Hardware Security without Secure Hardware: How to Decrypt with a Password and a Server

Olivier Blazy[1], Laura Brouilhet[1], Celine Chevalier[2], Patrick Towa[3], Ida Tucker[4], Damien Vergnaud[5,6]

[1] Université de Limoges, XLim, Limoges, France
[2] Université Panthéon-Assas Paris II, Paris, France
[3] ETH Zurich, Zurich, Switzerland
[4] IMDEA Software Institute, Madrid, Spain
[5] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[6] Institut Universitaire de France

Abstract. Hardware security tokens have now been used for several decades to store cryptographic keys. When deployed, the security of the corresponding schemes fundamentally relies on the tamper-resistance of the tokens – a very strong assumption in practice. Moreover, even secure tokens, which are expensive and cumbersome, can often be subverted.

We introduce a new cryptographic primitive called *Encryption schemes with Password-protected Assisted Decryption* (EPAD schemes), in which a user's decryption key is shared between a user device (or token) on which no assumption is made, and an online server. The user shares a human-memorizable password with the server. To decrypt a ciphertext, the user launches, from a public computer, a distributed protocol with the device and the server, authenticating herself to the server with her password (unknown to the device); in such a way that her secret key is never reconstructed during the interaction. We propose a strong security model which guarantees that (1) for an efficient adversary to infer any information about a user's plaintexts, it must know her password *and* have corrupted her device (secrecy is guaranteed if only one of the two conditions is fulfilled), (2) the device and the server are unable to infer any information about the ciphertexts they help to decrypt (even though they could together reconstruct the secret key), and (3) the user is able to verify that device and server both performed the expected computations. These EPAD schemes are in the password-only model, meaning that the user is not required to remember a trusted public key, and her password remains safe even if she is led to interact with a wrong server and a malicious device.

We then give a practical pairing-based EPAD scheme. Our construction is provably secure under standard computational assumptions, using non-interactive proof systems which can be efficiently instantiated in the standard security model, i.e., without relying on the random oracle heuristic.

1 Introduction

Mobile devices are ubiquitous nowadays: smart phones and tablets have not only become prevalent in daily communication but also in numerous security-critical tasks. These devices collect and compile a large amount of confidential information to which access must be controlled. If such a device is infected by malware, an attacker may gain full access to the compromised device and be able to control it and steal any information stored on it. In addition to that, smart phones and tablets are easily lost or stolen even though they usually only use (human-memorizable) passwords to prevent unauthorized access.

Despite their practicality, the use of mobile devices to store sensitive data incurs various security issues (since they fail to protect sensitive information and passwords in particular): vulnerability to dictionary attacks (since passwords are weak-entropy secrets), re-use of passwords for multiple services, frequent leakage of password databases, and many more. A possible solution is to use a physical device that provides extra security. A hardware security module is a tamper-resistant device that strengthens encryption practices and is used in addition to or in place of passwords. These modules often come in the form of plug-in cards or external devices from which secret information cannot be easily leaked to anyone who gets hold of it. However, such modules are often costly, inconvenient to use and may even be subverted, i.e., corrupt from their very production. In this paper, we investigate how we can get rid of such tamper-resistant devices, and show how to rely on a smart phone (combined with a public computer) on which no hardware-security assumption is made. We thus use a device (such as a smart phone) as a token, meaning in particular that these two terms are used interchangeably in the following.

Encryption schemes with Password-protected Assisted Decryption. In this work we focus on encryption, and consider the problem of achieving security guarantees equivalent to those of hardware security modules without making any assumption on the device in possession of the user. Therefore, no assumption is here made on a user token, i.e., it is not presumed to be tamper-proof or malware-free. The token just acts as a virtual smart-card.

To mitigate this lack of security from the token, we introduce a server which assists the user with the decryption. In a real-world scenario, the user could log-in from a public computer, use her phone as a token, and communicate with a remote server to decrypt ciphertexts she receives. Concretely, the secret key of the user is shared between the token and the server, and a password is shared between the user and the server. Nevertheless, the introduction of a server should not weaken the security of the original scheme: an attacker should not be able to leverage the server alone to infer any information about the plaintexts of the user. Besides, introducing a token and a server should not jeopardize the privacy of a user: they should assist her in a *blind* manner, i.e., without being able to infer any information about the plaintexts they help decrypt. This property is later referred to as *blindness*.

We also require *verifiability* for the user. First, although the user is not required to remember the decryption key, she should still be able to verify that the token and server both correctly performed their computations with respect to the public key. As we do not want to require the user to remember the public encryption key, it is assumed that a public key is attached to the ciphertext. Since a *different* key than that attached may have been used to compute the ciphertext, the protocol should only guarantee verifiability with respect to the attached key, but should not leak any information about the passwords held by the user and the server.

The advantage of having the user enter her password on a public computer rather than directly on her token is twofold. First, if an adversary takes control of the token, without knowing the user's password it cannot decrypt her ciphertexts (assuming an appropriate throttling mechanism preventing online dictionary attacks). Secondly, if the computer from which the user starts a decryption query is corrupt (e.g., has a keylogger on it), her password is leaked, but as long as her token is not corrupt, no one can decrypt her ciphertexts. Hence separating the user algorithm and the token guarantees security if either the user password is not leaked or if the token is not corrupt.

Three Levels of Authentication. For the sake of clarity, consider the user is logging-in from a public (untrusted) computer, and interacts with an (insecure) token (such as her smart phone or tablet) and a remote server. She shares a password with the server, and her decryption key is shared between the token and the server. Recall that no assumption is made on the hardware security of the token, but that she types her password into the public computer. We now present the authentication required between these three parties involved in our protocol.

User–Token Authentication. Since the token would in practice be a smart phone or a tablet, a PIN is usually required to access them (in very few attempts), which is similar to having the user authenticate to the token. The user can then initiate the decryption protocol between the three entities, from the public computer, by logging-in to the server using her password. No secure channel between the user's machine and the token is assumed.

Similarly, no higher-level mechanism is assumed for the token to authenticate itself to the user. The user is supposed to recognize her token and have it at proximity. Nonetheless, if the scheme is verifiable and secure, the user is assured that even if she is led to interact with a malicious token, the result of the decryption protocol must be correct if it terminates, and that the protocol leaks no information about her password.

Token–Server Authentication. The token authenticates itself to the server using its share of the user decryption key. Having the token prove to the server that it belongs to the user prevents adversaries from taking advantage of the servers' throttling mechanism to block an honest-user account. Indeed, without this authentication an attacker could request decryptions on behalf of the user with an

arbitrary token and make several password attempts until the server blocks her account. Even more damaging, the malicious token could also make queries to the server to infer information about the share of the server.

Likewise, the server must authenticate itself to the token using its own share of the decryption key. Otherwise, a malicious server without the user password could exploit the token to get information about it's share of the decryption key.

Server–User Authentication. In our model, the user need only remember her password. She must then be able to recover the address of the server sharing a decryption key with her token. A straightforward solution would be to retrieve this address from the token; but then a corrupt token may lure her into executing the decryption protocol with a malicious server via a phishing attack. The user may also simply mistype the address of the server. The server on which the user lands may even be certified within a PKI, but not one with which she shares a password, or one which shares the decryption key with the token. If this server is malicious, it could try to infer information about the user's password or the decryption key share held by the token.

Therefore, since the secret values (passwords or decryption-key shares) must be protected, the user and the server authenticate themselves to each other, and they do so via their common password. Note that this authentication must not leak any information on the password. In particular, it protects the user's password from an adversarial server which does not know her password, and it also preserves the confidentiality of her messages against an attacker which does not know her password and tries to exploit her server. These requirements are captured by our security definition.

A scheme satisfying all the aforementioned properties is called an Encryption scheme with Password-protected Decryption (EPAD) scheme.

Comparison with Prior Work. For cryptographic authentication, Camenisch, Lehmann, Neven and Samelin introduced [15] *password-authenticated server-aided signatures* (Pass2Sign). Their approach aims to offer comparable security guarantees to hardware security modules even when using a potentially corrupt device. To do so, they introduce a server which shares the secret key with the device. To compute a signature, the user starts a protocol with the server from her device, using a password to authenticate herself to the server. The secret key is never reconstructed during the protocol, so if the device is subsequently corrupted (assuming previously entered passwords have been erased), only a share of the secret key is lost and the attacker is unable to compute valid signatures. The device thus simply acts as a virtual smart card.

However, their work relies on two crucial assumptions: 1) the device is not corrupt at the moment the user enters her password, as an attacker would otherwise be able to impersonate the user and sign on her behalf and 2) the device securely erases previously typed passwords, since a corruption would directly leak them. Hence they do not completely achieve their ambitious goal of making no assumptions on the security of the device.

The scenario we consider (disregarding our new blindness property) is a hybrid of that of Pass2Sign (in which the user enters her password from the device) and of *password-protected secret sharing* [3, 4, 14, 30]. An important difference w.r.t. the latter is that *the decryption key is never reconstructed*, thereby protecting the user in case of corruption of her machine (i.e., the public computer from which she initiates the protocol). This property also allows the user to prevent further use of her device should it be stolen, by asking the server to block her account. This also hinders online dictionary attacks.

Moreover, interaction allows the server to enforce a throttling mechanism and refuse to decrypt if it detects suspicious behavior (e.g., several failed password attempts). From a commercial perspective, interaction also allows the server to run a paid service and charge the user for decryption requests.

Contrarily to the model of Pass2Sign, we do *not* assume that communication between the token and the server is a priori authenticated (via TLS for instance). Our security model ensures that interacting with a malicious party leaks no information about either the key shares or the passwords held by the user and the server. In our construction, the token and the server leverage the fact that they share the user's secret key to authenticate themselves to each other.

Finally, separating the public machine (on which the user types her password) and the token provides a *strictly stronger security* than that of password-authenticated server-aided signatures (for which a malware-infected token leaks both the user's password and secret-key share); while being no less convenient than technologies leveraging two-factor authentication mechanisms.

Contributions. The contributions of the paper are manifold.

Security Model. We first formalize the security properties required of an EPAD scheme. As explained above, an EPAD scheme enables a user to make decryption queries without revealing information about the ciphertexts being decrypted. This property was formalized by Green [26] (for classical public-key encryption (PKE) schemes) as blind decryption. To preserver user privacy, we propose a similar *blindness* property, the main difference being that there no longer is a centralized decryption entity as decryption is shared between the token and the server. Hence we require that neither the token nor the server should be able to infer any information about the ciphertexts the user wants to decrypt. The requirements of this property are strong, as it captures the scenario in which an adversary may have corrupted *both* the token and the server; and hence knows both shares of the decryption key, and the password. Our protocol achieves this strong notion of privacy by having the user perform some blinding step on the underlying plaintext (i.e., use a temporary high-entropy secret), and by randomizing the ciphertext before sending it to the token and the server for decryption. This implies that – to ensure user privacy – the scheme should tolerate some form of *malleability*.

Due to this mild form of malleability, which allows users to re-randomize their ciphertexts, the confidentiality notion considered for EPAD schemes is similar

to *Replayable Chosen-Ciphertext Attacks security* (RCCA) defined by Canetti, Krawczyk and Nielsen for classical PKE [18]. Our notion is called *Password-protected Indistinguishability under Replayable Chosen-Ciphertext Attacks* (P-IND-RCCA) and takes into account the fact that decryption requests should be *protected by user passwords*. More precisely, it ensures that unless an adversary both knows the user's password (by corrupting the user's machine or the server) *and* has corrupted her token, it cannot infer any information about the user's plaintexts in reasonable time. It captures both indistinguishability under replayable chosen-ciphertext attacks and password authentication. The formal model for P-IND-RCCA security is inspired by the Bellare–Rogaway–Pointcheval (BPR) model for password-based authenticated-key-exchange protocols [7]. It covers the cases of concurrent protocol executions, with potentially many users, tokens and servers.

The third security notion in our model is *verifiability* which guarantees that the user accepts the result of the decryption protocol only if the token and the server performed their computations correctly.

As we target the most efficient solutions possible, these properties are captured via game-based security definitions rather than functionalities in the universal composability framework [16]; see Section 4.2 for an in-depth discussion.

Technical Challenges. The fact user passwords are not entered into the token may, at first sight, seem a simple solution to the problem in Pass2Sign, and thereby to achieve the level of security provided by secure hardware. However, it raises several challenges. Indeed, (1) the token – without knowing the password – must be able to ensure that the user and server have correctly authenticated themselves to each other, and that it shares its secret key with the server, *before* performing computations. Otherwise, an attacker which does not know the password or the other key share, could exploit the token and gain information about its share. (2) Throughout the entire protocol, the parties must ensure they are communicating with the expected party, and that they are not victims of man-in-the-middle attacks (recall that the communication is not assumed to be a priori authenticated). (3) Although the token terminates decryption, plaintexts must remain hidden from its view, even though the user and server only share a low-entropy secret. (4) The token must be convinced that the server correctly performed its computation, even without knowing the only piece of information shared between the user and the server (i.e., the password). (5) Finally, the protocol should guarantee user privacy despite the fact that together, the token and the server know all secrets.

Efficient and Secure Construction. We overcome these challenges and propose a concrete pairing-based EPAD scheme. It uses as a building block the publicly verifiable RCCA-secure encryption scheme of Faonio, Fiore, Herranz and Ràfols [24], though similar techniques can be applied to other such schemes (e.g., Libert, Peters and Qian's [36]). Section 4.2 gives further insight into the technicalities of our scheme.

The construction may at first seem complex as it uses several classical cryptographic primitives as building blocks, and one may wonder whether a general Multi-Party Computation (MPC) between the token and the server could solve the problem. The issue is here that the token – without any information on the user's password – would need to verify the password held by the server. Hence a practical solution with off-the-shelf MPC is not immediate. Besides, it is not clear how such a solution would guarantee blindness.

Finally, our construction is proven secure in the standard model; one may wonder whether it could be simplified in a stronger model such as the Random-Oracle Model (ROM). However since our techniques heavily rely on the malleability of zero-knowledge proofs, ROM-based proofs do not seem appropriate.

2 Preliminaries

This section introduces the notation used throughout the paper, as well as the building blocks on which the constructions herein are based.

2.1 General Notation

The security parameter is denoted λ, and input lengths are always assumed to be bounded by some polynomial in λ. A Probabilistic algorithm is said to run in Polynomial Time (i.e., it is a PPT or efficient algorithm) if its running time is polynomial in λ.

Unless stated otherwise, p denotes a prime number. For a group $\mathbb{G}$ with neutral element $1_{\mathbb{G}}$, $\mathbb{G}^*$ stands for $\mathbb{G}\backslash\{1_{\mathbb{G}}\}$. For an integer $n \geq 1$, $[\![n]\!]$ denotes the set $\{1, \ldots, n\}$. Vectors and matrices are denoted in bold font, and vectors are by default column vectors. For a given matrix $\mathbf{A}$, $\mathbf{A}^{\mathrm{T}}$ stands for its transpose. For two matrices $\mathbf{A}$ and $\mathbf{B}$ of equal size, $\mathbf{A} \circ \mathbf{B}$ denotes their Hadamard (i.e., element-wise) product. If $\mathbf{A}$ and $\mathbf{B}$ are vectors, then their Hadamard product is simply denoted $\mathbf{AB}$.

Given an Oracle $\mathcal{O}$, the notation $(A, B, \cdot)$ means that (A, B) is its inner (secret) state, while $\cdot$ denotes the (adversarial) query.

Entropy is commonly used as a measure of password quality [13, 33], in this work the effort of guessing a password x sampled from some probability distribution $\mathcal{D}$ is measured by the min-entropy $H_{\infty,\mathcal{D}}(x)$. Min-entropy describes the unpredictability of an outcome determined solely by the probability of the most likely result, and is appropriate for describing passwords and other non-uniform distributions of secrets.

2.2 Bilinear Structures and SXDH Assumption

An *(asymmetric) bilinear structure* is a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ are p-order groups, and such that $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing, i.e., an efficiently computable non-degenerate ($e \neq 1_{\mathbb{G}_T}$) bilinear map. Let g_T denote $e(g_1, g_2)$, which is a generator of $\mathbb{G}_T$. Type-3 bilinear structures are bilinear

structures for which there is no known efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. A *bilinear structure generator* is an algorithm G which, on input 1^λ, returns the description of a bilinear structure. For integers $n, m \geq 1$, given vectors $\mathbf{x} \in \mathbb{G}_1^n$ and $\mathbf{y} \in \mathbb{G}_2^m$, $f(\mathbf{x}, \mathbf{y})$ denotes the matrix $\left[e\left(x_i, y_j\right)\right]_{i,j} \in \mathbb{G}_T^{n \times m}$.

We now introduce the hardness assumption on which our construction relies.

Definition 2.1 (SXDH Assumption). *The Symmetric eXternal Diffie–Hellman (SXDH) assumption over a bilinear structure generator* G *is that given* $\lambda \in \mathbb{N}$*, for* $\Gamma = (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$*, the Decisional Diffie–Hellman (DDH) assumption holds in both* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *with overwhelming probability. That is, no efficient adversary has a non-negligible advantage (in* λ*) in distinguishing* $\left(g_i, g_i^a, g_i^b, g_i^{ab}\right)$ *from* $\left(g_i, g_i^a, g_i^b, g_i^c\right)$*, for* $i \in \{1, 2\}$*,* $a, b, c \leftarrow_\$ \mathbb{Z}_p$ *and* $\Gamma \leftarrow \mathsf{G}\left(1^\lambda\right)$*.*

2.3 Signatures

We here introduce the syntax of digital signature schemes. See Appendix A for the definition of their strong one-time security and for specific instantiations.

Syntax & Correctness. A signature scheme consists of a setup algorithm $\mathsf{Setup}\left(1^\lambda\right) \rightarrow pp$, a key-generation algorithm $\mathsf{KG}(pp) \rightarrow (vk, sk)$, a signing algorithm $\mathsf{Sign}(sk, M) \rightarrow \sigma$ and a verification algorithm $\mathsf{Vf}(vk, M, \sigma) \rightarrow b \in \{0, 1\}$. The scheme is correct if $\mathsf{Vf}(vk, M, \mathsf{Sign}(sk, M)) = 1$ for all parameters and keys so generated, and all messages M.

2.4 Public-Key Encryption

This section introduces public-key encryption schemes, variants thereof and their security, as well as instantiations.

Syntax. A public-key encryption scheme consists of a setup algorithm $\mathsf{Setup}\left(1^\lambda\right) \rightarrow pp$, a key-generation algorithm $\mathsf{KG}(pp) \rightarrow (pk, sk)$ which returns a public encryption key and a secret decryption key, a probabilistic encryption algorithm $\mathsf{Enc}(pk, M; r) \rightarrow C$ (the randomness r may at times be omitted from the syntax) and a deterministic decryption algorithm $\mathsf{Dec}(sk, C) \rightarrow M/\perp$. An encryption scheme is *labeled* if the encryption and decryption algorithm additionally take as input a label or public data ℓ which is non-malleably attached to the ciphertext. In this case, the label is indicated on these algorithms by a superscript.

IND-PCA Security. INDistinguishability under Chosen *Plaintext-Checkable* Attacks [1] (IND-PCA) guarantees that an encryption scheme reveals no information about plaintexts even if an adversary can check whether ciphertexts encrypt messages of its choice. Abdalla, Benhamouda and Pointcheval [1] argued that this weakening of IND-CCA of security is enough for many password-related applications. In fact if the message space is small enough to enumerate all messages, it is equivalent to IND-CCA security. See Appendix B.1 for a formal definition.

RCCA Security. Indistinguishability under Replayable Chosen-Ciphertext Attacks [18] (RCCA) is a relaxation of the classical CCA security tolerating a mild form of malleability. It allows for the re-randomization of ciphertexts while still providing strong security guarantees. See Appendix B.1 for a formal definition.

Publicly Verifiable Encryption. A PKE scheme is verifiable if there exists a deterministic algorithm $\mathsf{Vf}(pk, C) \to b \in \{0,1\}$ such that no efficient adversary can, on the input of pk and with non-negligible probability, produce a ciphertext C such that $\mathsf{Vf}(pk, C) = 1$ and $\mathsf{Dec}(sk, C) = \perp$.

Re-randomizable Encryption. A PKE scheme is re-randomizable if there exists an algorithm $\mathsf{Rand}(pk, C) \to \hat{C}$ which given a public key pk and a ciphertext C, outputs a new ciphertext $\hat{C}$. It returns $\perp$ if any of its inputs are ill-formed.

Unlinkability. A re-randomizable encryption scheme is perfectly unlinkable [19, 36, 38] if the re-randomized valid ciphertexts have the same distribution as fresh encryptions of their underlying plaintexts.

Threshold Decryption. Threshold encryption schemes [20] are schemes in which decryption keys are shared between several parties. For a given ciphertext, each party can compute a decryption share with her key share, and a threshold number of those decryption shares is necessary to reconstruct the plaintext. If there are n parties, the security requirement of a t-out-of-n scheme is that no information about the plaintext can be inferred from less than $t+1$ shares. In the RCCA variant of this security notion, during the second query phase (which targets a specific honest party), the challenger first decrypts the ciphertext of the query with the secret key it has generated, and checks whether it results in one of the challenge messages before answering with a decryption share if it is not the case.

2.5 Smooth Projective Hash Functions

Smooth Projective Hash Functions (SPHFs) [21] are hash functions defined over a set $\mathcal{X}$, and which can be evaluated in two ways on a subset $\mathcal{L} \subseteq \mathcal{X}$. An SPHF can be evaluated on $\mathcal{X}$ using a hashing key hk, which can be seen as a private key. On $\mathcal{L}$, it can also be evaluated with a projective key hp, which can be seen a public key, and a witness of membership to $\mathcal{L}$. We use the definition due to Gennaro and Lindell [25], in which projective keys depend on words in $\mathcal{X}$.

Syntax [9]. An SPHF over a language $\mathcal{L} \subseteq \mathcal{X}$ is defined by five algorithms: $\mathsf{Setup}\left(1^{\lambda}\right) \to pp$ generates public parameters; $\mathsf{HashKG}(\mathcal{L}) \to hk$ generates a hashing key for $\mathcal{L}$; $\mathsf{ProjKG}(hk, \mathcal{L}, C) \to hp$ derives a projective key hp from hk depending on a word $C \in \mathcal{X}$; $\mathsf{Hash}(hk, \mathcal{L}, C) \to \mathfrak{H} \in \mathscr{H}$ outputs hash value for any word $C \in \mathcal{X}$, and $\mathsf{ProjHash}(hp, \mathcal{L}, C, w) \to \mathfrak{H} \in \mathscr{H}$ outputs a hash value on a word $C \in \mathcal{L}$ given a projective key and a witness w for the membership of C. The public parameters are given as implicit input to all the other algorithms. When $\mathcal{L}$ is the language of ciphertexts of a given message for a certain scheme, the public parameters typically contain the encryption key, and may even include the decryption key to efficiently test language membership.

Correctness. An SPHF is said to be correct if for all $C \in \mathcal{L}$ with witness w, for all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $hk \leftarrow \mathsf{HashKG}(\mathcal{L})$, $hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, C)$, $\mathsf{Hash}(hk, \mathcal{L}, C) = \mathsf{ProjHash}(hp, \mathcal{L}, C, w)$.

Adaptive Smoothness. An SPHF is smooth if its hash values on all $C \in \mathcal{X} \setminus \mathcal{L}$ are statistically indistinguishable from uniformly random values. Katz and Vaikuntanathan introduced [32] SPHFs (KV-SPHFs) with word-independent projective keys, and for which smoothness holds even if the words depend on the projective keys. KV-SPHFs are the most flexible kind since the words on which they are evaluated can be chosen even *after* computing and publishing the projective keys. In this sense, they are *adaptive*.

A KV-SPHF is smooth if its hash values on all $C \in \mathcal{X} \setminus \mathcal{L}$ are statistically indistinguishable from uniformly random values, even if C depends on projective keys. Formally, a KV-SPHF is ε-smooth [9] if, for any map $\mathfrak{f}$ onto $\mathcal{X} \setminus \mathcal{L}$, the following two distributions are ε-close:

$$\{(hp, \mathfrak{H}) : \ hk \leftarrow \mathsf{HashKG}(\mathcal{L}), hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \perp), \mathfrak{H} \leftarrow \mathsf{Hash}(hk, \mathcal{L}, \mathfrak{f}(hp))\}$$
$$\{(hp, \mathfrak{H}) : \ hk \leftarrow \mathsf{HashKG}(\mathcal{L}), hp \leftarrow \mathsf{ProjKG}(hk, \mathcal{L}, \perp), \mathfrak{H} \leftarrow_{\$} \mathscr{H}\}.$$

A KV-SPHF is *perfectly smooth* if it is 0-smooth.

2.6 Key-Derivation Functions

A Key-Derivation Function (KDF) computes pseudorandom keys of appropriate length from a source key material which is not uniformly distributed, or which still has high entropy despite partial adversarial knowledge. The results can then be used as secret keys for cryptosystems.

Syntax. A key-derivation function [35] $\mathsf{KDF}(SKM, XTS, CTX, L) \rightarrow K$, takes as input a source key material SKM, an extractor-salt value XTS, some context information CTX and a length L, and returns an L-bit string K. See Appendix E for the formal security definition of KDFs.

2.7 Malleable Non-Interactive Proofs

As the construction in Section 4 heavily relies on malleability and non-interactive zero-knowledge proofs, this section recalls the definition of proofs which are still sound under "controlled malleability". These proofs allow to compute, from a proof π on a word x, a new proof π' on a transformation $T_x(x)$ of x without the knowledge of a witness for $T_x(x)$, but only if the transformation belongs to a class of "allowed" transformations. The soundness of the proof system can then be defined w.r.t. this class of transformations. In addition to that, the soundness definition can even be extended to consider cases in which proofs are simulatable but remain sound under this controlled malleability.

Chase et al. [19] gave a definition of proof systems that are extractable under controlled malleability and a generic construction based on signatures and

extractable proof systems. In App. D.3, we give a similar definition which only requires soundness and then a generic construction from signatures and proof system that are only extractable in a sense defined therein. The reason is that the EPAD construction in Section 4 uses Groth–Sahai proofs from which group elements can be extracted but not exponents.

Syntax of Proof Systems. A non-interactive proof system for a language $\mathcal{L}$ (with corresponding relation $\mathcal{R}$) consists of an algorithm $\mathsf{Setup}\left(1^{\lambda}\right) \to crs$ which returns a common reference string, an algorithm $\mathsf{Prove}(crs, x, w) \to \pi$ which computes a proof on the input of a word x and of a witness w, and an algorithm $\mathsf{Vf}(crs, x, \pi) \to b \in \{0,1\}$ which returns a bit indicating whether the proof is considered valid. See App. D.3 for definitions of the classical soundness and and zero-knowledge properties.

Transformations. A *transformation* is an efficiently computable function $T := (T_x, T_w) : \mathcal{R} \to \mathcal{R}$. A relation $\mathcal{R}$ is said to be *closed* under T if for any $(x, w) \in \mathcal{R}$, $T(x, w) \in R$. Transformation T is then said to be *admissible* for $\mathcal{R}$. A class $\mathcal{T}$ of transformations is *allowable* for $\mathcal{R}$ if for every transformation $T \in \mathcal{T}$, T is admissible for $\mathcal{R}$.

A non-interactive proof system for a relation $\mathcal{R}$ is *malleable* [19] w.r.t. a class $\mathcal{T}$ of allowable transformations for $\mathcal{R}$ if there exists an algorithm $\mathsf{Eval}(crs, T, x, \pi) \to \pi'$ (word x may further be omitted from the syntax) which compute a proof π' for $T_x(x)$ from a valid proof π for x, without the knowledge of $T_w(w)$.

Groth–Sahai Proofs. Groth and Sahai [28] (GS) built an efficient proof system for a large class of equations in bilinear groups. It actually allows to extract witness group elements (but not exponents). Their proofs are re-randomizable and malleable w.r.t. additive transformations. The proof system is recalled in Appendix D.3. Using a structure-preserving signature scheme, one can apply the generic construction in Sec. D.1 to the GS proof system to obtain a zero-knowledge proof system which is CM simulation sound w.r.t. additive transformations.

3 Model

This section introduces Encryption schemes with Password-protected Assisted Decryption (EPAD schemes). As mentioned in the introduction, an EPAD scheme is an encryption scheme which involves three parties, a user $\mathcal{U}$, a token $\mathcal{T}$ and a server $\mathcal{S}$. The user $\mathcal{U}$, sharing a password with a server $\mathcal{S}$, is logging-in from a computer, and her decryption key is shared between the token $\mathcal{T}$ and the server $\mathcal{S}$. The protocol allows the user to decrypt ciphertexts with the help of the token $\mathcal{T}$ and the server $\mathcal{S}$, granted that each authentication between the three parties (based on the private values mentioned above) succeeds.

Informally, the security notions required for such a scheme are as follows. First, the *P-IND-RCCA* property captures both the indistinguishability of the

encryption scheme and password authentication. It ensures that the adversary cannot recover any information on the plaintext without knowing the password of the user and having access to the token, and that the decryption can only succeed if the user and the server share the same password. Then, the *blindness* property implies that neither the token nor the server can recover any information on the ciphertext the user wants to decrypt on their own. Finally, the *verifiability* property ensures that the user is convinced the decryption has been correctly done.

3.1 Syntax

In a three-party setting with a user $\mathcal{U}$, a token $\mathcal{T}$ and a server $\mathcal{S}$, an EPAD scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^{\lambda}\right) \to pp$: generates public parameters on input a security parameter. These parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}})$: generates a public key, a secret key and shares thereof.

$\mathsf{Enc}(pk, M) \to C$: a probabilistic encryption algorithm.

$\mathsf{Dec}(sk, C) \to M/\perp$: a deterministic decryption algorithm.

$\mathsf{IDec} = \langle \mathsf{U}(pk, p_{\mathcal{U}}, C) \rightleftharpoons \mathsf{T}(sk_{\mathcal{T}}) \rightleftharpoons \mathsf{S}(sk_{\mathcal{S}}, p_{\mathcal{S}}) \rangle \to \langle M/\perp, \perp, \perp \rangle$: an interactive decryption protocol between a user algorithm with input a public key, a user password and a ciphertext; a token algorithm with input a secret-key share; and a server algorithm with input a secret-key share and a server password. The passwords are here treated as bit strings.

Correctness. An EPAD scheme is correct if the decryption of an encrypted plaintext, whether by the deterministic decryption algorithm or by the interactive protocol with $p_{\mathcal{U}} = p_{\mathcal{S}}$, results in the plaintext. That is, for all $\lambda \in \mathbb{N}$, all M and all $p_{\mathcal{U}} = p_{\mathcal{S}}$,

$$\Pr\left[\mathsf{IDec} = \langle \mathsf{Dec}(sk, C), \perp, \perp \rangle = \langle M, \perp, \perp \rangle : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ C \leftarrow \mathsf{Enc}(pk, M) \end{array}\right] = 1.$$

3.2 Security Definitions

This section formalizes the security properties expected from EPAD schemes. These properties are Password-protected Indistinguishability under Replayable Chosen-Ciphertext Attacks (P-IND-RCCA), blindness and verifiability.

P-IND-RCCA Security. Password-protected Indistinguishability under Replayable Chosen-Ciphertext Attacks (P-IND-RCCA) ensures that no efficient adversary can infer any information about a user's plaintext as long as it does not know her password (by corrupting the user's machine or the server) or does not

have access to her token. It captures both indistinguishability under replayable chosen-ciphertext attacks and password authentication. The latter means that decryption can only succeed if the user and the server have the same password.

The formal model for P-IND-RCCA security is inspired by the Bellare–Rogaway–Pointcheval (BPR) model for password-based authenticated-key-exchange protocols [7]. It covers the cases of concurrent protocol executions, with potentially many users, tokens and servers. In addition to that, a user may possess several tokens and could be registered on several servers.

Game Overview. The P-IND-RCCA security experiment features an adversary $\mathcal{A}$. After an initial parameter generation phase, $\mathcal{A}$ can request that the challenger generates keys and passwords for the users. The passwords are generated via a password generator PG that returns values in some dictionary D. The min-entropy of the output distribution of PG is denoted $H_{\infty,\mathsf{PG}}$.

Next, $\mathcal{A}$ is given access to several oracles modeling different types of attacks (password-related or chosen-ciphertext attacks); and to a test oracle which can be called at any time, but only once (a definition with several test queries would be equivalent). On input two messages and a user identity chosen by $\mathcal{A}$, this test oracle randomly chooses one of the messages, and returns an encryption of this message under the user's public key. If $\mathcal{A}$ guesses which message was encrypted, it is considered successful. An EPAD scheme is then said to be P-IND-RCCA secure if no efficient adversary can win the game with probability significantly greater than $1/2+\epsilon$, where ϵ is the maximum advantage one can gain from trivial online dictionary attacks (i.e., guessing passwords should be the only possible attacks).

Initialization & Game Variables. A set of users U is assumed to be fixed. For every $\mathcal{U} \in U$, the set of tokens belonging to $\mathcal{U}$ is denoted by $T[\mathcal{U}]$, and the set of servers with which $\mathcal{U}$ shares a password is denoted $S[\mathcal{U}]$. The set of all tokens is denoted T and the set of all servers S.

During the initialization phase, public parameters for the encryption scheme are generated. Secret-key shares for all tokens and servers are set to $\perp$. Further on, for $id \in T \cup S$, sk_{id} denotes the set of all user key-shares for id.

A finite instance set I for all party algorithms is also assumed to be fixed. Each instance $i \in I$ of the algorithm of party id maintains a state st^i_{id}. A session identifier sid^i_{id}, and partner identities pid^i_0 and pid^i_1 allow to match instances in protocol executions.

A variable $used^i_{id}$ indicates whether an active attack has been performed on the ith instance of the algorithm of party id.

Variables acc^i_{id} and $term^i_{id}$ respectively indicate whether the ith algorithm instance of party id has accepted and terminated. As in the BPR model, acceptance and termination are distinguished. When an instance terminates, it does not output any further message. Nevertheless, it can accept at a certain point of its computation but terminate later. This may occur when an instance confirms its partners, in which case it accepts, and thereafter continues the computation until termination.

A queue Q_{AddU} of added users, i.e., users for which keys and passwords have been generated, and a queue of corrupt users Q_{Corrupt} are also initialized.

At the end of the initialization phase, the encryption parameters, the sets of participants and the user public keys are returned in a public input pin, and the rest is set in a secret input sin. That is, $pin \leftarrow (pp,\ I,\ U,\ T,\ S,\ (pk_{\mathcal{U}})_{\mathcal{U}})$ and $sin \leftarrow \big(pin, (sk_{\mathcal{T}})_{\mathcal{T}}, (sk_{\mathcal{S}})_{\mathcal{S}}, \big(st^i_{id}, sid^i_{id}, pid^i_{id,0}, pid^i_{id,1}, \ acc^i_{id}, term^i_{id}, used^i_{id}\big)_{i,id}, Q_{\mathsf{AddU}},\ Q_{\mathsf{Corrupt}}\big)$. The secret input sin is later made available to all oracles.

Oracles. Throughout the experiment, $\mathcal{A}$ is given access to the oracles detailed below and summarized on Figure 1, and which it can query in any order.

- Test_b : returns the encryption with a user public key of one of two messages, all chosen by the adversary. The challenge user identity $\mathcal{U}^*$ is not required to be honest (i.e., the adversary may know her password), but the challenge token $\mathcal{T}^*$ and challenge server $\mathcal{S}^*$ cannot both be corrupt, otherwise the adversary would be able to reconstruct her secret key and trivially win the game. The public key $pk_{\mathcal{U}^*}$ is the one for which $\mathcal{T}^*$ and $\mathcal{S}^*$ hold secret-key shares. For simplicity, the adversary may query this oracle at most once. Note also that as in the BPR model [7] and the model for distributed session key [8], this query is not restricted to be the last query of the adversary.
- AddU : adds an honest user identity. In addition to a user identity $\mathcal{U}$, the adversary specifies a set $T[\mathcal{U}]$ of tokens and a set $S[\mathcal{U}]$ of servers for $\mathcal{U}$. Note that these can be corrupt, except for the challenge identities $\mathcal{U}^*$, $\mathcal{T}^*$, $\mathcal{S}^*$: parties $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt (see the definition of oracle $\mathsf{Corrupt}$). For each server in $S[\mathcal{U}]$, a password $p_{\mathcal{U}}$ and a transformation $p^{\mathcal{U}}_{\mathcal{S}}$ thereof is generated by the password generator PG. Keys and secret-key shares for all token–server pairs of the user are also generated.
- Exec : returns the transcript of an honest (i.e., without the interference of the adversary) decryption-protocol execution on a ciphertext C. Note that Exec queries thereby model offline dictionary attacks among others. The execution is between the ith, jth and kth instances the algorithms of a user $\mathcal{U}$, a token $\mathcal{T}$ and a server $\mathcal{S}$. The notation U_i, T_j and S_k mean that algorithms U, T and S are respectively run with the states $st^i_{\mathcal{U}}$, $st^j_{\mathcal{T}}$ and $st^k_{\mathcal{S}}$. If $\mathsf{Dec}(sk_{\mathcal{U}}, C)$ is one of the challenge messages (with $sk_{\mathcal{U}}$ the secret key for which $\mathcal{T}$ and $\mathcal{S}$ hold shares), the oracle returns a special string replay as in the classical definition of RCCA security (see Sec. 2.4). The parties may be corrupt (in which case $\mathcal{A}$ has their states), but $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt.
- Send : the adversary can perform active attacks via this oracle. The adversary can send a message to an algorithm instance (e.g., the kth instance of a server algorithm), all of its choice. The notation $\mathsf{IDec}(id, \cdot)$ respectively stands for $\mathsf{U}(\cdot)$, $\mathsf{T}(\cdot)$ or $\mathsf{S}(\cdot)$ if $id \in U$, T or S. This algorithm then runs the instance on that message. To prompt the ith instance of the algorithm of a user $\mathcal{U}$ to initiate a protocol execution on a ciphertext C with the jth instance of a token $\mathcal{T}$ and the kth instance of a server $\mathcal{S}$, the adversary can query

$\mathsf{Init}\left(1^\lambda, U, T, S, I\right)$	$pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$ for $(\mathcal{T} \in T)\left\{\text{for } (\mathcal{U} \in U)\ sk_{\mathcal{T}}^{\mathcal{U}} \leftarrow \perp\right\}$ for $(\mathcal{S} \in S)\left\{\text{for } (\mathcal{U} \in U)\ sk_{\mathcal{S}}^{\mathcal{U}} \leftarrow \perp\right\}$ for $(i, id) \in I \times (U \cup T \cup S)$ $\quad st_{id}^i \leftarrow \perp$ $\quad sid_{id}^i \leftarrow pid_{id,0}^i \leftarrow pid_{id,1}^i \leftarrow \perp$ $\quad acc_{id}^i \leftarrow term_{id}^i \leftarrow used_{id}^i \leftarrow \mathsf{FALSE}$ $Q_{\mathsf{AddU}} \leftarrow Q_{\mathsf{Corrupt}} \leftarrow \emptyset$ return (pin, sin)
$\mathsf{AddU}(\mathcal{U}, T[\mathcal{U}], S[\mathcal{U}])$	if $(\mathcal{U} \notin U$ or $\mathcal{U} \in Q_{\mathsf{Corrupt}})$ return $\perp$ for $(\mathcal{S} \in S[\mathcal{U}])\left(p_{\mathcal{U}}, p_{\mathcal{S}}^{\mathcal{U}}\right) \leftarrow \mathsf{PG}(pp)$ // Generate passwords for the user for $(\mathcal{T}, \mathcal{S}) \in T[\mathcal{U}] \times S[\mathcal{U}]$ $\quad (pk_{\mathcal{U}}, sk_{\mathcal{U}}, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp)$ // Generate user keys $\quad \left(sk_{\mathcal{T}}^{\mathcal{U}}, sk_{\mathcal{S}}^{\mathcal{U}}\right) \leftarrow (sk_{\mathcal{T}}, sk_{\mathcal{S}})$ $\quad$ output $pk_{\mathcal{U}}$ $Q_{\mathsf{AddU}} \leftarrow Q_{\mathsf{AddU}} \cup \{\mathcal{U}\}$
$\mathsf{Exec}(\mathcal{U}, i, \mathcal{T}, j, \mathcal{S}, k, C)$	if $(\mathcal{U} \notin Q_{\mathsf{AddU}}$ or $\mathcal{T} \notin T$ or $\mathcal{S} \notin S)$ $\quad$ return $\perp$ if $\left(used_{\mathcal{U}}^i \text{ or } used_{\mathcal{T}}^j \text{ or } used_{\mathcal{S}}^k\right)$ return $\perp$ if $(\mathsf{Dec}(sk_{\mathcal{U}}, C) \in \{M_0, M_1\})$ return replay // Prevent replay attacks $\tau \leftarrow \left\langle \mathsf{U}_i(pk, p_{\mathcal{U}}, C), \mathsf{T}_j\left(sk_{\mathcal{T}}^{\mathcal{U}}\right), \mathsf{S}_k\left(sk_{\mathcal{S}}^{\mathcal{U}}, p_{\mathcal{S}}^{\mathcal{U}}\right)\right\rangle$ // Run an honest protocol execution. return τ

$\mathsf{Send}(id, i, M)$	if $term_{id}^i$ return $\perp$
	$used_{id}^i \leftarrow \mathsf{TRUE}$
	
	if $id = \mathcal{U}^*$ and $M = (\mathcal{T}^*, *, \mathcal{S}^*, *, C)$
	$\quad$ if $(\mathsf{Dec}(sk_{\mathcal{U}^*}, C) \in \{M_0, M_1\})$
	$\quad\quad$ return replay
	if $\mathcal{U}^* \in Q_{\mathsf{Corrupt}}$ or $\mathcal{S}^* \in Q_{\mathsf{Corrupt}}$
	$\quad$ if $(id = \mathcal{T}^*)$ return $\perp$
	// If $\mathcal{A}$ has the password of $\mathcal{U}^*$, reject queries to $\mathcal{T}^*$
	
	if $\mathcal{U}^* \in Q_{\mathsf{Corrupt}}$ and $\mathcal{T}^* \in Q_{\mathsf{Corrupt}}$
	$\quad$ if $(id = \mathcal{S}^*)$ return $\perp$
	// If $\mathcal{A}$ has the password of $\mathcal{U}^*$ and if $\mathcal{T}^*$ is corrupt, reject queries to $\mathcal{S}^*$
	
	$\left\langle m_{out}, acc, term_{id}^i, sid, pid_0, pid_1, st_{id}^i\right\rangle \leftarrow \left\langle \mathsf{IDec}\left(id, st_{id}^i, sk_{id}, p_{id}, M\right)\right\rangle$
	// If $id \in U$, replace sk_{id} with pk_{id}
	// If $id \in T$, replace p_{id} with $\perp$
	
	if acc and $\neg acc_{id}^i$
	$\quad sid_{id}^i \leftarrow sid;\ pid_{id,0/1}^i \leftarrow pid_{0/1};\ acc_{id}^i \leftarrow acc$
	return $\left(m_{out}, acc, term_{id}^i, sid, pid_0, pid_1, st_{id}^i\right)$

$\mathsf{Corrupt}(id, p)$	if $\left(\exists i \in I\colon used_{id}^i \text{ and } \neg term_{id}^i\right)$ return $\perp$ //Static corruption only $Q_{\mathsf{Corrupt}} \leftarrow Q_{\mathsf{Corrupt}} \cup \{id\}$ if $id \in U$ $\quad$ for $\mathcal{S} \in S[id]$ $\quad\quad p_{\mathcal{S}}^{\mathcal{U}} \leftarrow p[\mathcal{S}]$ // Overwrite the user's passwords $\quad$ return $\left(p_{\mathcal{U}}, \left\{st_{\mathcal{U}}^i\right\}_{i \in I}\right)$ else $\quad$ if $\left(id = \mathcal{T}^* \text{ and } \mathcal{S}^* \in Q_{\mathsf{Corrupt}}\right)$ or $\left(id = \mathcal{S}^* \text{ and } \mathcal{T}^* \in Q_{\mathsf{Corrupt}}\right)$ $\quad\quad$ return $\perp$ // $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt $\quad$ return $\left(sk_{id}, \left\{st_{id}^i\right\}_i\right)$
$\mathsf{Test}_b\left(\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*, M_0, M_1\right)$	if $(\mathcal{U}^* \notin U$ or $\mathcal{T}^* \notin T[U^*]$ or $\mathcal{S}^* \notin S[U^*])$ return $\perp$ if $\left((\mathcal{T}^*, \mathcal{S}^*) \in Q_{\mathsf{Corrupt}}^2\right)$ return $\perp$ // $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt $C^* \leftarrow \mathsf{Enc}\left(pk_{\mathcal{U}^*}, M_b\right)$ return C^*

Fig. 1. Oracles for the P-IND-RCCA Security Experiment.

oracle Send on $(\mathcal{U}, i, (\mathcal{T}, j, \mathcal{S}, k, C))$. If such a query decrypt to either of the challenge messages, the oracle returns replay to prevent trivial wins.

In addition to an output message m_{out}, the algorithm also returns to $\mathcal{A}$ acceptance and termination states acc and $term^i_{id}$, a session identifier sid and partner instances pid_0 and pid_1, and a state st^i_{id}. Identity pid_0 is always assumed to be the party which should receive the next flow of id, i.e., a token identity if $id \in U$, a server identity if $id \in T$ and a token identity if $id \in S$. Variables sid^i_{id}, $pid^i_{id,0}$ and $pid^i_{id,1}$ and acc^i_{id} are updated in case the instance accepts, and all values are revealed to the adversary except the state (which may contain a password or a secret-key share). $\mathcal{A}$ can access this state by corrupting party id.

If $\mathcal{A}$ knows the password of the challenge user $\mathcal{U}^*$, by corrupting either that identity or $\mathcal{S}^*$, then queries to $\mathcal{T}^*$ are rejected. It translates the fact that in case of a server breach or if there is a keylogger on the user's machine, then no security can be expected if an attacker also has access to her token.

Likewise, if $\mathcal{U}^*$ is corrupt as well as $\mathcal{T}^*$, queries to $\mathcal{S}^*$ are rejected. It reflects the fact that if a user's token is corrupt, no security can be expected if her password is also leaked. Indeed, in that case, an attacker can use the server which shares a key with the token in order to decrypt the ciphertexts.

– Corrupt : gives the adversary control over all the instances of a party algorithm. In the case of a user, the adversary does not only receives her passwords and the states of all her algorithm instances, but may also overwrite the password held by each of her servers. If the party being corrupted is a token or a server, the adversary receives the states of all of its algorithm instances and also its key shares. Once a Test query has been made with an identity $\mathcal{U}^*$, the corruption of both $\mathcal{T}^*$ and $\mathcal{S}^*$ is not allowed. That is to prevent the adversary from reconstructing her secret key and trivially win the game.

  Only *static corruptions* are here considered, i.e., the adversary cannot corrupt a party of which an algorithm instance is in the middle of a protocol execution. It is expressed by the condition "if $(\exists i \in I\colon used^i_{id}$ and $\neg term^i_{id})$ return $\perp$".

Definition 3.1 (P-IND-RCCA). *An EPAD scheme is P-IND-RCCA secure if for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l} \\ b = b'\colon \\ \\ \end{array}\begin{array}{l} (pin, sin) \leftarrow \mathsf{Init}\left(1^\lambda, U, T, S, I\right) \\ b \leftarrow_\$ \{0,1\} \\ \mathcal{O} \leftarrow \{\mathsf{Exec}, \mathsf{Send}, \mathsf{Corrupt}, \mathsf{Test}_b\} \\ b' \leftarrow \mathcal{A}^{\mathcal{O}(sin,\cdot)}(pin) \\ return\ (b, b') \end{array}\right]$$

is negligibly close to $1/2 + q_{\mathsf{Send}} \cdot H_{\infty,\mathsf{PG}}$.

The term $q_{\mathsf{Send}} \cdot H_{\infty,\mathsf{PG}}$ accounts for online dictionary attacks. An EPAD scheme should guarantee that these are the best attacks possible.

Remark 3.1 (On One-Round Protocols). Note that there cannot exist a one-round protocol secure against offline dictionary attacks. A round is here understood as the set of messages sent between all parties from a message sent by the user to the token to the next response from the token. Would there be such a one-round protocol, an adversary could intercept the transcript of an execution of the decryption protocol. In that execution, the user would have had to send sufficient information for the server to verify her password, without having received any prior indication that the server shares this password; sufficient in fact, for the adversary to perform an offline dictionary attack on her password. Therefore, a protocol secure against offline dictionary attacks must consist of at least two rounds. It is not surprising as the user, who initiates the protocol, should verify the server holds the same password as she does. Moreover, the mere fact the user continues the protocol until the end indicates that the server could successfully authenticate itself to the user. On this account, without a throttling mechanism on the user's side, online dictionary attacks against the user's password cannot be prevented.

Blindness. This property formalizes the idea that neither the token nor the server should be able to infer any information about the ciphertexts the user wants to decrypt. This is analogous to Green's notion of blindness [26] except that in his work, the decryptor is a centralised entity, whereas the decryption process is here shared between the token and the server.

In the formal definition, the challenge ciphertexts C_0 and C_1 must be either both valid or both invalid, i.e., $(\mathsf{Dec}(sk, C_0) = \perp) = (\mathsf{Dec}(sk, C_1) = \perp)$ should hold, which is a minimal condition to exclude trivial wins. Besides, the definition might at first seem too strict as the token and server are therein corrupt and can together reconstruct the full secret key and the password. However, the definition can in practice be achieved by letting the user rerandomize her ciphertexts and blind the underlying plaintexts with temporary high-entropy secrets before sending the ciphertext to the token and the server.

Definition 3.2 (Blindness). *An EPAD scheme satisfies blindness if for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l} b = b' \colon \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ (st, p_{\mathcal{U}}, C_0, C_1) \leftarrow \mathcal{A}(pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \\ b \leftarrow_{\$} \{0,1\} \\ \textit{if } (\mathsf{Dec}(sk, C_0) = \perp) \neq (\mathsf{Dec}(sk, C_1) = \perp) \\ \quad b' \leftarrow_{\$} \{0,1\} \\ \quad \textit{return } (b, b') \\ \langle *, b' \rangle \leftarrow \langle \mathsf{U}(pk, p_{\mathcal{U}}, C_b), \mathcal{A}(st) \rangle \\ \textit{return } (b, b') \end{array}\right]$$

is negligibly close to 1/2.

Verifiability. This property captures the idea that the user should accept the result of the decryption protocol only if the token and the server have correctly performed their computations.

Definition 3.3 (Verifiability). *An EPAD scheme is verifiable if for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,*

$$\Pr\left[\langle \mathsf{U}\left(pk, p_{\mathcal{U}}, C\right), \mathcal{A}(st)\rangle \notin \{\langle \mathsf{Dec}(sk, C), *\rangle, \langle \perp, *\rangle\} : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathsf{KG}(pp) \\ (st, p_{\mathcal{U}}, C) \leftarrow \mathcal{A}(pk, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \end{array}\right] = 1.$$

4 Construction

In this section, we build an EPAD scheme from the RCCA-secure encryption scheme of Faonio, Fiore, Herranz and Ràfols [24] (see App. B.3). Similar techniques can a priori be applied to any publicly verifiable, structure-preserving RCCA-secure scheme. The public-verifiability aspect is to easily make the scheme threshold while maintaining the security of the original scheme.

The section starts by showing how users can blind the plaintexts underlying the ciphertexts they want to decrypt. It then continues with the main construction and its efficiency assessment.

4.1 Verification of Blinded Ciphertexts

As mentioned in Sec. 3.2, EPAD schemes should satisfy blindness, meaning that even if the token and the server are corrupt, they cannot infer any information about the ciphertexts they are helping a user to decrypt. It is a stringent requirement as a corrupt token and server can reconstruct the secret key, and if the server is actually one associated to the user, it also has her password. Thus given a ciphertext, the user must be able to blind the underlying plaintext using only public information.

On the other hand, EPAD schemes also aim for an RCCA type of security, so the only type of malleability that should be expected is re-randomization. Therefore, the user cannot a priori blind the plaintext and produce a new valid ciphertext (except with negligible probability) since she does not remember her secret key. Nonetheless, she can provide enough information *auxiliary* to the modified ciphertext which allows the token and the server to verify, *with their secret key shares*, that she correctly blinded the underlying plaintext of a valid ciphertext, and that she knows the blinding factor.

We thus introduce a new algorithm $\mathsf{Blind}(pk, C) \rightarrow \left(\tilde{C}, aux\right)$ which, given a public key and a ciphertext of the scheme of Faonio et al., essentially adds a one-time pad to the plaintext and gives a Groth–Sahai proof knowledge of the pad in some auxiliary information. The new ciphertext can then be verified by another algorithm $\mathsf{BlindVf}\left(sk, \tilde{C}, aux\right) \rightarrow b \in \{0, 1\}$ on the input of a secret-key

share and of the auxiliary information. Similar techniques can a priori be applied to other publicly-verifiable schemes. Algorithms Blind and BlindVf are formally defined in App. B.3.

4.2 Construction

This section presents our main construction which is further denoted $\mathcal{E}$. Recall from Sec. 3 that a secure scheme must be at least two rounds. Our protocol, which consists of two rounds, is therefore round optimal.

Building Blocks. The construction uses as building blocks

- the RCCA-secure encryption scheme of Faonio et al. (App. B.3) denoted $\mathcal{E}_{\mathsf{rcca}}$
- the short Cramer–Shoup encryption scheme in $\mathbb{G}_1$ (App. B.2) with hash-function family $\mathcal{H}_{\mathrm{PCA}}$ to encrypt passwords. It is further denoted $\mathcal{E}_{\mathsf{pca}}$
- Groth's one-time signature scheme (see App. A.1), further denoted OTS, with hash-function family $\mathcal{H}_{\mathrm{OTS}}$
- the KV-SPHF for short Cramer–Shoup ciphertexts (App. C.2)
- a (single-keyed) Hash-based Message Authentication Code [6] (HMAC), further denoted MAC, with $\{g_\kappa\}_\kappa$ as family of compression functions (SHA-256 in practice)
- Krawczyk's KDF (App. E) denoted KDF with
  - for the extraction phase, HMAC based on a family $\{H_\kappa\}_\kappa$ of Merkle–Damgård hash functions with $\{h_\kappa\}_\kappa$ as underlying family of compression functions (SHA-512 in practice), and
  - for the expansion phase, HMAC with $\{g_\kappa\}_\kappa$ as family of compression functions (SHA-256 in practice)
- the SXDH-based simulation-sound Groth–Sahai proof system (App. D.3) which is controllably malleable w.r.t. additive transformations in $\mathbb{Z}_p$ and satisfies strong derivation privacy (App. D.2), and which uses Jutla and Roy's scheme (further denoted SIG) as underlying signature scheme (App. A.2). The proof system is further denoted SS_GS.

Construction Overview. The main steps of the scheme are depicted on Fig 2. The high-level description follows (for notations and details, see the appendices cited in the building blocks above).

Setup & Key Generation. The parameters include parameters for the scheme of Faonio et al., for the SPHF for Elgamal ciphertexts, and for the simulation-sound GS proof system.

Encryption & Decryption Algorithms. These are the same as the ones of the scheme of Faonio et al.

Interactive Decryption. During the interactive decryption protocol IDec, the parties proceed as follows.

– At the beginning of the protocol, the user and the server essentially do a one-round Password-Authenticated Key Exchange (PAKE) with short Cramer–Shoup encryption of their passwords and KV-SPHF evaluations on them; following techniques of Benhamouda et al. [9]. The underlying idea is to enable each party to implicitly check via the projective keys that the ciphertext of the other party encrypts the same password as hers (see Sec. 2.5).

  The encrypted passwords are bound to the ongoing session via the projective keys sent during the PAKE and used as labels for the encrypted passwords. It prevents replay attacks since the corresponding hashing keys are needed to recover the PAKE key.

  The key obtained later serves two purposes. It is first used as key material for a KDF of which the output is used to authenticate the respective next flows of the server and then the user, thereby making sure that the other party holds the same password. Its second use is to mask the partial decryption of the server so that only a party who has the same password as the server can later remove it and retrieve the plaintext.

  In parallel, the token and the server verify that the other party knows a share of the secret key, and the user checks that the token and the server can together reconstruct the secret key. The user therefore makes sure that the server knows *both* a secret-key share, and her password if their PAKE outputs are the same (which she ascertains with the MAC).

  Note that, to bind its proof to the ongoing session, the token also signs the proof (and the input from the user) with a one-time scheme. The verification key is used as label for an encryption (with a key different from the one used to encrypt the passwords) of the partial user public key relative to the token share. As only the server can also compute the partial user public key of the token, it can check (given the decryption key) that the token is the party who computed the ciphertext, and that the one-time verification key was not altered.

  On the other hand, the server need not do the same as no computation involving the token share is done before the server must prove knowledge of the hashing key corresponding to the projective key sent in the first round. As the projective key is authenticated together with the server proof via the MAC, and as the smoothness of the SPHF guarantees that the hashing key can be recovered with only negligible probability, the server proof is also bound to the ongoing session.
– After the PAKE, the user re-randomizes her ciphertext and blinds the underlying plaintext (as in Sec. 4.1) so that even if the token and the server are corrupt, they cannot infer any information about the ciphertext they help her decrypt. She also authenticates the blinded, re-randomized ciphertext with a key derived with the KDF from the PAKE key as key material. She then sends the ciphertext and the tag to the token.

– The token verifies that the user correctly blinded the plaintext of a valid ciphertext, and that she knows the blinding factor, before forwarding to the server what it just received. This verification could actually be done *in parallel* of the server computation, but before partially decrypting the ciphertext. It is to make sure that no attacker can obtain partial decryption from the token on invalid ciphertexts and possibly infer information about its share.
– The server verifies the authenticity of the flow via the MAC and then performs the same verifications as the token. If they succeed, it partially decrypts the ciphertext and masks it with the PAKE key. The server then sends the partially decrypted (and masked) ciphertext to the token, along with a proof that she decrypted the ciphertext with the key share of which she proved knowledge *in the first round* (i.e., verification is done w.r.t. the GS commitment to that share and which was sent in the first round), and that it masked the result with the PAKE key from the first round. Although the token does *not* know the password shared by the user and the token, it is convinced that the mask is really the PAKE key from the first round. That is because the token verifies the server proof w.r.t. the encrypted passwords and the projective keys that were sent in the first round, and this is absolutely crucial to prevent man-in-the-middle attacks between the token and the server.
– After verifying the proof, the token finishes the decryption and uses the malleability of GS proofs w.r.t. additive transformations to compute, from the server proof, a proof that it and the server both correctly performed their computations. It then sends the decrypted, though blinded (by the user) and masked, plaintext and the proof of correct computation.
– The user verifies the proof, and if it is correct, removes her blinding factor, removes the mask with the PAKE key and can then recover the plaintext.

In the P-IND-RCCA security proof, soundness must be guaranteed even after giving the adversary simulated proofs, which is why the GS proof system must be CM simulation sound w.r.t. additive transformations (see Section D.1). It is due to the fact that secret keys are kept across different sessions.

Formal Description. The decryption protocol is given on Fig. 3. Each message sent is assumed to be prepended with a session identifier and the identities of the two partner instances. It is assumed that an algorithm aborts if it receives an ill-formed message or if a verification fails, and that it erases all its temporary variables (which include its randomness) once it terminates. The proofs from the token and the server are outlined below.

Notation. Let $[\cdot]\colon \mathbb{G}_1 \to \{0,1\}^\ell$ be an injection, namely the bit representation of $\mathbb{G}_1$ elements. Integers in $\{0,\ldots,p-1\}$ are identified with their binary representations in $\{0,1\}^\ell$. For three integers $i \in \{1,2\}$, j and k, whenever computational

Parameter Generation: Generate key pair (ek, sk) for IND-PCA secure PKE. Publish ek, discard sk.

Key Generation: Generate $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$, and shares $(sk_{\mathcal{T}}, sk_{\mathcal{S}})$ of $sk_{\mathcal{U}}$ for IND-RCCA secure PKE.

Interactive Decryption of C_M (for user password $p_{\mathcal{U}}$ and server password $p_{\mathcal{S}}$):

$\mathsf{U}(pk_{\mathcal{U}}, p_{\mathcal{U}}, C_M)$		$\mathsf{T}(sk_{\mathcal{T}})$		$\mathsf{S}(sk_{\mathcal{S}}, p_{\mathcal{S}})$
- Generate $hk_{\mathcal{U}}$: user hashing key and $hp_{\mathcal{U}}$: associated projection key - Compute $C_{\mathcal{U}}$: IND-PCA encryption of $p_{\mathcal{U}}$ w/ randomness $r_{\mathcal{U}}$	$\xrightarrow{C_{\mathcal{U}}, hp_{\mathcal{U}}}$	- Compute $\tilde{\pi}_{\mathcal{T}}$: malleable ZKPoK of $sk_{\mathcal{T}}$ proving $sk_{\mathcal{T}}$ is a share of $sk_{\mathcal{U}}$ - Generate a ciphertext $C_{\mathcal{T}}$: allows to check consistency with $sk_{\mathcal{S}}$ w.r.t. $pk_{\mathcal{U}}$ - One-Time Sign $(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}})$ to get $\sigma_{\mathcal{T}}$	$\xrightarrow[\sigma_{\mathcal{T}}]{(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}})}$	- Verify signature $\sigma_{\mathcal{T}}$ - Verify proof $\tilde{\pi}_{\mathcal{T}}$ - Check $sk_{\mathcal{T}}$ consistent w/ $sk_{\mathcal{S}}$ using $C_{\mathcal{T}}$ - Use malleability of $\tilde{\pi}_{\mathcal{T}}$ to compute $\tilde{\pi}_{\mathcal{S}}$: ZKP that $(sk_{\mathcal{T}}, sk_{\mathcal{S}})$ are shares of $sk_{\mathcal{U}}$ - Generate $hk_{\mathcal{S}}$: server hash key and $hp_{\mathcal{S}}$: associated projection key - Compute $C_{\mathcal{S}}$: IND-PCA encryption of $p_{\mathcal{S}}$ w/ randomness $r_{\mathcal{S}}$
- Verify proof $\tilde{\pi}_{\mathcal{S}}$ - Compute $h'_{\mathcal{U}}$: hash of $C_{\mathcal{U}}$ w/ proj key $hp_{\mathcal{S}}$ & witness $r_{\mathcal{U}}$ - Compute $h'_{\mathcal{S}}$: hash of $C_{\mathcal{S}}$ w/ hash key $hk_{\mathcal{U}}$ - Compute $K_{\mathcal{U}} \leftarrow \mathsf{KDF}(h'_{\mathcal{U}} \cdot h'_{\mathcal{S}})$ - Verify $\tau_{\mathcal{S}} = \mathsf{MAC}$ of $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}})$ w/ key $K_{\mathcal{U}}$ - Randomize C_M to get $\tilde{C}_M$ - Blind plaintext in $\tilde{C}_M$ to get $\tilde{C}_M$ and proof π_{B} of correct blinding & knowledge of blinding factor - Compute $\tau_{\mathcal{U}}$: MAC of $(\tilde{C}_M, \pi_{\mathsf{B}})$ w/ key $K_{\mathcal{U}}$	$\xleftarrow[\tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}}]{C_{\mathcal{S}}, hp_{\mathcal{S}}}$	- Verify proof $\tilde{\pi}_{\mathcal{S}}$	$\xleftarrow[\tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}}]{C_{\mathcal{S}}, hp_{\mathcal{S}}}$	- Compute $h_{\mathcal{U}}$: hash of $C_{\mathcal{U}}$ w/ hash key $hk_{\mathcal{S}}$ - Compute $h_{\mathcal{S}}$: hash of $C_{\mathcal{S}}$ w/ proj key $hp_{\mathcal{U}}$ & witness $r_{\mathcal{S}}$ - Compute $K_{\mathcal{S}} \leftarrow \mathsf{KDF}(h_{\mathcal{U}} \cdot h_{\mathcal{S}})$ - Compute $\tau_{\mathcal{S}}$: MAC of $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}})$ w/ key $K_{\mathcal{S}}$
	$\xrightarrow[\tau_{\mathcal{U}}]{\tilde{C}_M, \pi_{\mathsf{B}}}$	- Verify proof π_{B} with $sk_{\mathcal{T}}$	$\xrightarrow[\tau_{\mathcal{U}}]{\tilde{C}_M, \pi_{\mathsf{B}}}$	- Verify $\tau_{\mathcal{U}} = \mathsf{MAC}$ of $(\tilde{C}_M, \pi_{\mathsf{B}})$ w/ key $K_{\mathcal{S}}$ - Verify π_{B} with $sk_{\mathcal{S}}$
- Verify proof $\pi_{\mathcal{T}}$ - Compute $M \leftarrow K_{\mathcal{U}}^{-1} M_{\mathcal{T}}$ - Return M	$\xleftarrow{M_{\mathcal{T}}, \pi_{\mathcal{T}}}$	- Verify proof $\pi_{\mathcal{S}}$ - Partially decrypt $M_{\mathcal{S}}$ w/ $sk_{\mathcal{T}}$ to get $M_{\mathcal{T}}$ - Use malleability of $\pi_{\mathcal{S}}$ to compute $\pi_{\mathcal{T}}$: ZKP that $M_{\mathcal{S}} = \mathsf{Dec}(sk_{\mathcal{U}}, \tilde{C}_M)$	$\xleftarrow{M_{\mathcal{S}}, \pi_{\mathcal{S}}}$	- Partially decrypt $\tilde{C}_M$ w/ $sk_{\mathcal{S}}$ to get $\bar{M}_{\mathcal{S}}$ - Mask $\bar{M}_{\mathcal{S}}$ to get $M_{\mathcal{S}} = K_{\mathcal{S}} \cdot \bar{M}_{\mathcal{S}}$. - Compute $\pi_{\mathcal{S}}$: malleable ZKPoK of $(hk_{\mathcal{S}}, p_{\mathcal{S}}, r_{\mathcal{S}})$ proving that $hp_{\mathcal{S}}$, $C_{\mathcal{S}}$, $M_{\mathcal{S}}$ are honestly computed.

Fig. 2. Construction Overview

costs in a bilinear structure are considered, $\mathbb{G}_i^j$ denotes a j-exponentiation in $\mathbb{G}_i$ and P^k denotes the computation of product k pairing values. As for communication costs, $j\mathbb{G}_i$ denotes j elements in $\mathbb{G}_i$ and $j\mathbb{Z}_p$ denotes j elements in $\mathbb{Z}_p$.

In the following SS_GS is a Simulation-Sound Groth Sahai proof system with two main algorithms, a Prove algorithm that generates valid proofs for a statement with respect to the secret used, and Vf that checks if a proof is valid with respect to a statement. An extra algorithm Eval allows to further refine a proof by using extra witnesses.

OTS is a one time signature, with a KG key generation algorithm, a Sign signing algorithm, and a verification Vf algorithm checking the consistency of the signature with respect to it's public key.

$\mathcal{E}_{\mathsf{pca}}$ is a plaintext checkable encryption scheme, while $\mathcal{E}_{\mathsf{rcca}}$ is a randomizable CCA encryption scheme allowing access to extra algorithms: a randomization Rand algorithm, and Blind/BlindVf that respectively allow to blind the payload in a ciphertext, and to check consistency to know whether the decryption algorithm will return a value different from $\perp$.

HashKG, ProjKG, Hash, ProjHash are the 4 algorithms constituting an SPHF, to generate keys to allow to implicitly prove the validity of a statement with respect to a given witness.

Parameters. Given two families $\mathcal{H}_{\mathrm{PCA}}$ and $\mathcal{H}_{\mathrm{OTS}}$ of hash functions from $\{0,1\}^*$ to $\{0,1\}^\ell$, to generate public parameters,

- generate a bilinear structure $\Gamma \leftarrow (p, \mathbb{G}_1 = \langle g_1\rangle, \mathbb{G}_2 = \langle g_2\rangle, \mathbb{G}_T, e) \leftarrow_\$ \mathsf{G}\left(1^\lambda\right)$. Let $\ell \leftarrow \lfloor \log_2 p \rfloor + 1$ be the bit length of p
- select $H_{\mathrm{PCA}} \leftarrow_\$ \mathcal{H}_{\mathrm{PCA}}$ and $H_{\mathrm{OTS}} \leftarrow_\$ \mathcal{H}_{\mathrm{OTS}}$
- generate a salt value $XTS \leftarrow_\$ \{0,1\}^\ell$ for the KDF
- generate keys for the short Cramer–Shoup encryption scheme in $\mathbb{G}_1$, i.e.,
  Decryption key $dk = (\zeta, \alpha, \beta, \alpha', \beta')$
  Encryption key $ek = (h_1, \gamma, \delta) = \left(g_1^\zeta,\ g_1^\alpha h_1^\beta,\ g_1^{\alpha'} h_1^{\beta'}\right)$
- generate parameters for $\mathcal{E}_{\mathsf{rcca}}$, i.e., GS parameters $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$; $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$ in soundness mode
- generate parameters for SS_GS, i.e., GS parameters $\tilde{\mathbf{a}}, \tilde{\mathbf{b}} \in \mathbb{G}_1^2$; $\tilde{\mathbf{v}}, \tilde{\mathbf{w}} \in \mathbb{G}_2^2$ in soundness mode and a pair of keys $(vk, sk) \leftarrow \mathsf{SIG.KG}(\Gamma, 2)$
- return $pp \leftarrow crs \leftarrow \left(\Gamma; \mathbf{a}, \mathbf{b}; \mathbf{v}, \mathbf{w}; \tilde{\mathbf{a}}, \tilde{\mathbf{b}}; \tilde{\mathbf{v}}, \tilde{\mathbf{w}}; H_{\mathrm{PCA}}, ek; H_{\mathrm{OTS}}; vk; XTS\right)$.

Passwords. Passwords are elements of $\mathbb{G}_1$; $1_{\mathbb{G}_1}$ is not a valid password.

Key Generation. To generate keys for $\mathcal{E}$, run $(pk_{\mathcal{U}}, sk, sk_{\mathcal{T}}, sk_{\mathcal{S}}) \leftarrow \mathcal{E}_{\mathsf{rcca}}.\mathsf{KG}(pp)$. The secret-key shares $sk_{\mathcal{T}}/sk_{\mathcal{S}} \leftarrow (\alpha_{1,\mathcal{T}/\mathcal{S}},\ \alpha_{2,\mathcal{T}/\mathcal{S}},\ \mathbf{G},\ A_1,\ A_2,\ \mathbf{C}_1,\ \mathbf{C}_2)$, are such that $\alpha_{i,\mathcal{T}} \leftarrow_\$ \mathbb{Z}_p$ and $\alpha_{i,\mathcal{S}} = \alpha_i - \alpha_{i,\mathcal{T}}$ for $i = 1, 2$. Generate also another pair $(ek', dk') \leftarrow \mathcal{E}_{\mathsf{pca}}.\mathsf{KG}(p, g_1, \mathbb{G}_1, H)$. Note that $pk_{\mathcal{U}}$ is of the form $\big(\Gamma, pk_1 \coloneqq g_1^D,$ $pk_2 \coloneqq g_1^{\alpha_1 D + \alpha_2}, \ldots\big)$. The token is given ek', and the server dk'. The whole token

$$crs \coloneqq \left(\Gamma; \mathbf{a}, \mathbf{b}; \mathbf{v}, \mathbf{w}; \tilde{\mathbf{a}}, \tilde{\mathbf{b}}; \tilde{\mathbf{v}}, \tilde{\mathbf{w}}; H, ek; H_{\mathrm{OTS}}; vk; XTS\right)$$

$\mathsf{U}(pk_{\mathcal{U}}, p_{\mathcal{U}}, C_M)$		$\mathsf{T}(sk_{\mathcal{T}}, ek')$		$\mathsf{S}(sk_{\mathcal{S}}, p_{\mathcal{S}}, dk')$
$hk_{\mathcal{U}} \leftarrow \mathsf{HashKG}(\mathcal{L}_{p\mathcal{U}})$	$\xrightarrow{C_{\mathcal{U}}, hp_{\mathcal{U}}}$	$\tilde{\pi}_{\mathcal{T}} \leftarrow \mathsf{SS\_GS.Prove}\left\{\alpha_{i,\mathcal{T}} \colon pk_2 pk_1^{-\alpha_{1,\mathcal{S}}} g_1^{-\alpha_{2,\mathcal{S}}} = pk_1^{\alpha_{1,\mathcal{T}}} g_1^{\alpha_{2,\mathcal{T}}}\right\}$	$\xrightarrow{C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk}$ $\tilde{\pi}_{\mathcal{T}}, \sigma_{\mathcal{T}}, C_{\mathcal{T}}$	$\mathsf{OTS.Vf}\left(ovk, \left(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}\right), \sigma_{\mathcal{T}}\right) \stackrel{?}{=} 1$
$hp_{\mathcal{U}} \leftarrow \mathsf{ProjKG}\left(hk_{\mathcal{U}}, \mathcal{L}_{p\mathcal{U}}, \perp\right)$				$\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}\left(dk', C_{\mathcal{T}}\right) \stackrel{?}{=} pk_2 pk_1^{-\alpha_{1,\mathcal{S}}} g_1^{-\alpha_{2,\mathcal{S}}}$
$C_{\mathcal{U}} \leftarrow \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{hp_{\mathcal{U}}}\left(ek, p_{\mathcal{U}}; r_{\mathcal{U}}\right)$		$(ovk, osk) \leftarrow \mathsf{OTS.KG}(p, \mathbb{G}_1, H_{\mathrm{OTS}})$		$\mathsf{SS\_GS.Vf}\left(crs, pk_{\mathcal{U}}, \alpha_{i,\mathcal{S}}, \tilde{\pi}_{\mathcal{T}}\right) \stackrel{?}{=} 1$
		$C_{\mathcal{T}} \leftarrow \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}\left(ek', pk_1^{\alpha_{1,\mathcal{T}}} g_1^{\alpha_{2,\mathcal{T}}}\right)$		$\tilde{\pi}_{\mathcal{S}} \leftarrow \mathsf{SS\_GS.Eval}\left(crs, \alpha_{i,\mathcal{S}}, \tilde{\pi}_{\mathcal{T}}\right)$
		$\sigma_{\mathcal{T}} \leftarrow \mathsf{OTS.Sign}\left(osk, \left(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}\right)\right)$		
				$hk_{\mathcal{S}} \leftarrow \mathsf{HashKG}\left(\mathcal{L}_{p\mathcal{S}}\right)$
				$hp_{\mathcal{S}} \leftarrow \mathsf{ProjKG}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, \perp\right)$
				$C_{\mathcal{S}} \leftarrow \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{hp_{\mathcal{S}}}\left(ek, p_{\mathcal{S}}; r_{\mathcal{S}}\right)$
				$H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, C_{\mathcal{U}}\right)$
				$\cdot\, \mathsf{ProjHash}\left(hp_{\mathcal{U}}, \mathcal{L}_{p\mathcal{S}}, C_{\mathcal{S}}, r_{\mathcal{S}}\right)$
$\mathsf{SS\_GS.Vf}\left(crs, pk_{\mathcal{U}}, \tilde{\pi}_{\mathcal{S}}\right) \stackrel{?}{=} 1$				
$H_{\mathcal{U}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p\mathcal{U}}, C_{\mathcal{S}}\right)$				$K_{\mathcal{S}} \leftarrow \mathsf{KDF}\left([H_{\mathcal{S}}], XTS, \perp, \ell\right)$
$\cdot\, \mathsf{ProjHash}\left(hp_{\mathcal{S}}, \mathcal{L}_{p\mathcal{U}}, C_{\mathcal{U}}, r_{\mathcal{U}}\right)$				
$K_{\mathcal{U}} \leftarrow \mathsf{KDF}\left([H_{\mathcal{U}}], XTS, \perp, \ell\right)$	$\xleftarrow{C_{\mathcal{S}}, hp_{\mathcal{S}}}$ $\tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}}$	$\mathsf{SS\_GS.Vf}\left(crs, sk_{\mathcal{T}}, \tilde{\pi}_{\mathcal{S}}\right) \stackrel{?}{=} 1$	$\xleftarrow{C_{\mathcal{S}}, hp_{\mathcal{S}}}$ $\tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}}$	$\tau_{\mathcal{S}} \leftarrow \mathsf{MAC}\left(K_{\mathcal{S}}, \left(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}}\right)\right)$
$\mathsf{MAC}\left(K_{\mathcal{U}}, \left(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}}\right)\right) \stackrel{?}{=} \tau_{\mathcal{S}}$				
$\hat{C}_M \leftarrow \mathcal{E}_{\mathsf{rcca}}.\mathsf{Rand}(pk_{\mathcal{U}}, C_M)$				$\mathsf{MAC}\left(K_{\mathcal{S}}, \left(\tilde{C}_M, aux\right)\right) \stackrel{?}{=} \tau_{\mathcal{U}}$
$\left(\tilde{C}_M, aux\right) \leftarrow \mathcal{E}_{\mathsf{rcca}}.\mathsf{Blind}\left(pk, \hat{C}_M\right)$	$\xrightarrow{\tilde{C}_M, aux}$ $\tau_{\mathcal{U}}$	$\mathcal{E}_{\mathsf{rcca}}.\mathsf{BlindVf}\left(sk_{\mathcal{S}}, \tilde{C}_M, aux\right) \stackrel{?}{=} 1$	$\xrightarrow{\tilde{C}_M, aux}$ $\tau_{\mathcal{U}}$	$\mathcal{E}_{\mathsf{rcca}}.\mathsf{BlindVf}\left(sk_{\mathcal{T}}, \tilde{C}_M, aux\right) \stackrel{?}{=} 1$
$\tau_{\mathcal{U}} \leftarrow \mathsf{MAC}\left(K_{\mathcal{U}}, \left(\tilde{C}_M, aux\right)\right)$				$M_{\mathcal{S}} \leftarrow \tilde{x}_3 \tilde{x}_1^{-\alpha_{1,\mathcal{S}}} \tilde{x}_2^{-\alpha_{2,\mathcal{S}}} H_{\mathcal{S}}$
				$\pi_{\mathcal{S}} \leftarrow \mathsf{SS\_GS.Prove}\{hk_{\mathcal{S}}, p_{\mathcal{S}}, r_{\mathcal{S}} \colon$
				$hp_{\mathcal{S}} = \mathsf{ProjKG}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, \perp\right)$
$\mathsf{SS\_GS.Vf}\left(\ldots, M_{\mathcal{T}}, \tilde{\pi}_{\mathcal{S}}, \pi_{\mathcal{T}}\right) \stackrel{?}{=} 1$		$\mathsf{SS\_GS.Vf}\left(crs, C_{\mathcal{U}}, C_{\mathcal{S}}, hp_{\mathcal{U}}, \tilde{C}_M, M_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \pi_{\mathcal{S}}\right) \stackrel{?}{=} 1$		$C_{\mathcal{S}} = \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{hp_{\mathcal{S}}}(ek, p_{\mathcal{S}}; r_{\mathcal{S}})$
$M \leftarrow H_{\mathcal{U}}^{-1} M_{\mathcal{T}}$	$\xleftarrow{M_{\mathcal{T}}, \pi_{\mathcal{T}}}$	$M_{\mathcal{T}} \leftarrow M_{\mathcal{S}} \tilde{x}_1^{-\alpha_{1,\mathcal{T}}} \tilde{x}_2^{-\alpha_{2,\mathcal{T}}}$	$\xleftarrow{M_{\mathcal{S}}, \pi_{\mathcal{S}}}$	$M_{\mathcal{S}} \tilde{x}_3^{-1} = \tilde{x}_1^{-\alpha_{1,\mathcal{S}}} \tilde{x}_2^{-\alpha_{2,\mathcal{S}}} \cdot \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p\mathcal{S}}, C_{\mathcal{U}}\right)$
return M		$\pi_{\mathcal{T}} \leftarrow \mathsf{SS\_GS.Eval}(crs, \alpha_{i,\mathcal{T}}, \pi_{\mathcal{S}})$		$\cdot\, \mathsf{ProjHash}\left(hp_{\mathcal{U}}, \mathcal{L}_{p\mathcal{S}}, C_{\mathcal{S}}, r_{\mathcal{S}}\right)\}$

Fig. 3. Decryption Protocol.

share (cf. the syntax of Sec. 3) is then $(sk_{\mathcal{T}}, ek')$ and the whole server share is $(sk_{\mathcal{S}}, dk')$.

Acceptance and Termination. The U instance accepts after verifying tag $\tau_{\mathcal{S}}$, and terminates after returning M. The T instance accepts after receiving $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$, and terminates after sending $M_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to the user. The S instance accepts after verifying $\tau_{\mathcal{U}}$, and terminates after sending $M_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$ to the token.

First Proofs $\tilde{\pi}_{\mathcal{T}}$ from the Token. In the first round, the token proves with $\mathsf{SS\_GS}$ to the server that it knows the other share of the user secret key. Proof $\tilde{\pi}_{\mathcal{T}}$ then consists of 20 $\mathbb{G}_1$ element, 26 $\mathbb{G}_2$ elements and 2 $\mathbb{Z}_p$ elements, and verifying it costs $4\mathbb{G}_1^1 + 4\mathbb{G}_2^1 + 2\mathrm{P}^4 + 6\mathrm{P}^6 + 8\mathrm{P}^5 + 4\mathrm{P}^8$.

First Proofs $\tilde{\pi}_{\mathcal{S}}$ from the Server. Leveraging the malleability of $\mathsf{SS\_GS}$, the server computes a proof of knowledge of the complete user secret key.

Second Proof $\pi_{\mathcal{S}}$ from the Server. After the server partially decrypts the ciphertext and masks the result with an SPH, it proves that it correctly performed its computation. In particular, the server proves the server proves that secret-key shares it used to partially decrypt are the ones to which it committed *in the first round*. It also proves that the password it encrypted in the first round is the same that is used to computed the SPH mask.

Equations. Let $\xi \leftarrow H(U, E, hp_{\mathcal{S}})$. The server must prove knowledge of $p_{\mathcal{S}}$, $hk_{\mathcal{S}} \coloneqq (\lambda, \mu, \nu, \theta)$ and $r_{\mathcal{S}}$ such that

$$hp_{\mathcal{S},1} = g_1^{\lambda} h_1^{\nu} \gamma^{\theta} \text{ and } hp_{\mathcal{S},2} = g_1^{\mu} \delta^{\theta} \text{ and } C_{\mathcal{S}} = (U, E, V) = \left(g_1^{r_{\mathcal{S}}}, h_1^{r_{\mathcal{S}}} p_{\mathcal{S}}, \left(\gamma \delta^{\xi}\right)^{r_{\mathcal{S}}}\right)$$
$$\text{and } M_{\mathcal{S}} \tilde{x}_3^{-1} = \tilde{x}_1^{-\alpha_{1,\mathcal{S}}} \tilde{x}_2^{-\alpha_{2,\mathcal{S}}} U^{\lambda+\mu\xi} E^{\nu} (1/p_{\mathcal{S}})^{\nu} V^{\theta} \left(hp_{\mathcal{U},1} hp_{\mathcal{U},2}^{\xi}\right)^{r_{\mathcal{S}}}.$$

With $\mathsf{SS\_GS}$, proof $\pi_{\mathcal{S}}$ consists of 38 $\mathbb{G}_1$ elements, 42 $\mathbb{G}_2$ elements and 2 $\mathbb{Z}_p$ elements. Verifying it costs $4\mathbb{G}_1^1 + \mathbb{G}_2^2 + 2\mathrm{P}^3 + 4\mathrm{P}^4 + 4\mathrm{P}^5 + 6\mathrm{P}^6 + 6\mathrm{P}^7 + 4\mathrm{P}^9 + 8\mathrm{P}^{10} + 2\mathrm{P}^{13}$. Note that the token verifies w.r.t. to the commitments $\mathbf{c}(\alpha_{i,\mathcal{S}})$ it infers from the first proof $\tilde{\pi}_{\mathcal{S}}$ and its secret-key share.

Second Proof $\pi_{\mathcal{T}}$ from the Token. Upon receiving $\pi_{\mathcal{S}}$ from the server, the token decryption algorithm first verifies it. If $\pi_{\mathcal{S}}$ is correct, the token algorithm uses the malleability of GS proofs to compute a proof that decryption was done with the full secret key of which knowledge was proved in the first round.

Correctness & Security. The P-IND-RCCA security and the verifiability of $\mathcal{E}$ rely of the following assumptions (see also App. E):

- $\{h_\kappa\}_\kappa$ is pairwise-independent
- $\{H_\kappa\}_\kappa$ is collision-resistant against linear-size circuits
- $g.$ is a secure Pseudo-Random Function (PRF)

	$\mathsf{U} \rightleftharpoons \mathsf{T}$	$\mathsf{T} \rightleftharpoons \mathsf{S}$
1^{st} Flow	$5\mathbb{G}_1$	$31\mathbb{G}_1 + 26\mathbb{G}_2 + 4\mathbb{Z}_p$
2^{nd} Flow	$25\mathbb{G}_1 + 26\mathbb{G}_2 + 3\mathbb{Z}_p$	$25\mathbb{G}_1 + 26\mathbb{G}_2 + 3\mathbb{Z}_p$
3^{rd} Flow	$16\mathbb{G}_1 + 19\mathbb{G}_2 + 4\mathbb{G}_T + 1\mathbb{Z}_p$	$16\mathbb{G}_1 + 19\mathbb{G}_2 + 4\mathbb{G}_T + 1\mathbb{Z}_p$
4^{th} Flow	$39\mathbb{G}_1 + 42\mathbb{G}_2 + 1\mathbb{Z}_p$	$39\mathbb{G}_1 + 42\mathbb{G}_2 + 1\mathbb{Z}_p$

Table 1. Theoretical communication cost of the decryption protocol.

	$\mathsf{U} \rightleftharpoons \mathsf{T}$	$\mathsf{T} \rightleftharpoons \mathsf{S}$
1^{st} Flow	0.41	5
2^{nd} Flow	4.43	4.43
3^{rd} Flow	4.92	4.92
4^{th} Flow	6.73	6.73

Table 2. Estimated communication in KB, $\lambda = 128$ and Cocks-Pinch modified curve.

– $\hat{g}\colon (K, M) \to g_M(K)$ is a secure PRF under a class of affine related-key attacks defined by the inner and outer pads [5, Lemma 5.2].

Concerning its properties, $\mathcal{E}$ is correct. Assuming that the SXDH assumption over G holds, that $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g_\cdot$ and $\hat{g}_\cdot$ satisfy the assumptions above, that $\mathcal{H}_{\text{PCA}}$ is second-preimage resistant and that $\mathcal{H}_{\text{OTS}}$ is collision-resistant, $\mathcal{E}$ is P-IND-RCCA secure. Moreover, $\mathcal{E}$ satisfies blindness under the SXDH assumption over G. Lastly, $\mathcal{E}$ is verifiable if $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g_\cdot$ and $\hat{g}_\cdot$ satisfy the assumptions above and if the SXDH assumption over G holds. See App. F for proofs of these statements.

Efficiency. Tab. 1 sums up the theoretical communication cost of the decryption protocol. Concretely, for an 128-bit security, adopting the Cocks-Pinch modified curve with p a 672 bit prime and an embedding degree $k = 6$ (following parameters advised by Guillevic in [29]), elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are of size 672 bits, while elements of $\mathbb{G}_T$ are of size 4028 bits. We estimate the resulting communication complexity in Tab. 2.

On Adaptive Corruptions. The main reason why the decryption protocol is not secure against adaptive corruptions is that the short Cramer–Shoup encryption is "fully committing", meaning that there is only one valid opening (i.e., message–randomness pair) for each commitment (i.e., ciphertext). However, in one of the intermediate games in the proof of P-IND-RCCA security, the challenger computes commitments to dummy passwords for honest parties. It means that if the adversary corrupts an honest party right after she has sent her commitment, the challenger cannot return to the adversary a valid opening which contains the actual password of that party.

To overcome this hurdle, one could instead use an equivocable commitment scheme which supports adaptively smooth KV-SPHFs. As an equivocable commitment scheme allows to compute a valid opening to any commitment given a trapdoor, such a scheme together with a KV-SPHF should make the protocol secure against adaptive corruptions. Blazy and Chevalier's commitment scheme and its associated KV-SPHF [10] precisely satisfy these conditions. The commitment scheme relies on the SXDH assumption, and although its size is constant in the bit length of the message (i.e., 4 $\mathbb{G}_1$ elements and 2 $\mathbb{G}_2$ elements for each commitment), it is still larger than that of the short Cramer–Shoup encryption scheme. The latter was then chosen for efficiency reasons but at the expense of adaptive corruptions.

Note also that if $\mathcal{U}^*$ is corrupted right after the end of the PAKE (i.e., at the end of the first round), then the adversary could get from the server instance a partial decryption on the challenge ciphertext with the third flow (even if the $\mathcal{U}^*$ instance was prompted on a different ciphertext). To prevent this, the token could again compute a one-time signature as in the first flow, but also include C_M in the label of the encryption. Doing so ensures the server that the third flow went through the token, which would not be possible in case of corruption of $\mathcal{U}^*$ as token queries are then rejected.

On Composability. The Section-3 definitions are game-based and do not guarantee composability. However, a UC [16] functionality for EPAD schemes could be defined in the same vein as for PAKE [17] and RCCA-secure encryption schemes [18] considered together. Yet, to achieve composability, and as for PAKE protocols, it would be necessary for the UC simulator (in case the adversary correctly guesses the password of an honest party) to be able to compute correct SPH values on *invalid* words C (encrypting $1_{\mathbb{G}_1}$) with the sole knowledge of a projective key hp from the adversary and not of the corresponding hashing key hk.

Benhamouda et al. [9] introduced trapdoor SPHFs exactly for this purpose and gave a construction for the original Cramer–Shoup encryption scheme which is computationally and adaptively smooth under the SXDH assumption. It could readily be adapted to the short version of their scheme, though each projective key would contain one more $\mathbb{G}_1$ element. Alternatively, one could also use structure-preserving SPHFs which were introduced by Blazy and Chevalier [10].

Mitigating Server Breaches. EPAD schemes have so far been defined only w.r.t. a single server. Nevertheless, password theft from server databases is common in practice, and users even tend to use the same password for several services. It means that not only the confidentiality of user messages is threatened if a single server is compromised (and if her token is corrupt), but also potential other services.

To mitigate the impact of server breaches, a potential solution is to use threshold cryptography [22,39]. User passwords are then encrypted in a database and the decryption key is shared between $n \geq 1$ servers so that no information

about the passwords is leaked if at most a number t of them are corrupt. Nevertheless, any t+1 servers should be able to recover user passwords. Blazy, Chevalier and Vergnaud [11, Sec. 5] proposed an efficient protocol for threshold PAKE [37] from SPHFs which is based on this idea. It allows a user and a gateway that interacts with $t+1$ servers to agree on a common high-entropy key if the user password and the encrypted password match, without revealing any information about the passwords. However, if they do not, the keys obtained by each party are independent and uniformly random. In consequence, the security of the key exchange is guaranteed so long as at most t servers are corrupt.

The EPAD scheme in Section 3 can then be turned into a scheme with n servers and which withstand the corruption of t of them (i.e., a t-out-of-n scheme) as follows. During the set-up phase, the user encrypts her password and sends the encrypted password and a t-out-of-n secret-key share to each server. The $\mathcal{E}_{\mathsf{rcca}}$ secret key is $(t+1)$-out-of-$(n+1)$ shared between the n servers and the token so that any $t+1$ servers and the token can decrypt user ciphertexts. During the decryption protocol, each of the participating servers plays in parallel the role of the gateway of Blazy, Chevalier and Vergnaud's protocol, and uses the resulting keys to authenticate the second flow in the protocol of Fig. 3. In the last flow, each server would then have to prove that it partially decrypted the ciphertext with its $\mathcal{E}_{\mathsf{rcca}}$ key share and masked it with the key resulting from the threshold PAKE. As the threshold PAKE is based on SPHFs, the same techniques (as in Sec. 3) leveraging malleability apply.

Acknowledgements

This work was supported by the French ANR Projects ALAMBIC (ANR-16-CE39-0006), IDFIX (ANR-16-CE39-0004) and the EU H2020 Research and Innovation Program under Grant Agreement No. 786725 (OLYMPUS). Most of the work was done while the fourth author was at IBM Research – Zurich and ENS and PSL Research University.

References

1. M. Abdalla, F. Benhamouda, and D. Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 332–352. Springer, Heidelberg, Mar. / Apr. 2015.
2. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, Aug. 2009.
3. M. Abdalla, M. Cornejo, A. Nitulescu, and D. Pointcheval. Robust password-protected secret sharing. In I. G. Askoxylakis, S. Ioannidis, S. K. Katsikas, and C. A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 61–79. Springer, Heidelberg, Sept. 2016.
4. A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. Password-protected secret sharing. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM CCS 2011*, pages 433–444. ACM Press, Oct. 2011.

5. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, Aug. 2006.
6. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, Aug. 1996.
7. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.
8. M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995.
9. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, Aug. 2013.
10. O. Blazy and C. Chevalier. Structure-preserving smooth projective hashing. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, Dec. 2016.
11. O. Blazy, C. Chevalier, and D. Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 3–18. Springer, Heidelberg, Feb. / Mar. 2016.
12. O. Blazy, D. Pointcheval, and D. Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, Heidelberg, Mar. 2012.
13. J. Bonneau. Statistical metrics for individual password strength. In *Security Protocols Workshop*, 2012.
14. J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 256–275. Springer, Heidelberg, Aug. 2014.
15. J. Camenisch, A. Lehmann, G. Neven, and K. Samelin. Virtual smart cards: How to sign with a password and a server. In V. Zikas and R. De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 353–371. Springer, Heidelberg, Aug. / Sept. 2016.
16. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
17. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.
18. R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, Aug. 2003.
19. M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, Apr. 2012.
20. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001.

21. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, Apr. / May 2002.
22. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, Aug. 1990.
23. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.
24. A. Faonio, D. Fiore, J. Herranz, and C. Ràfols. Structure-preserving and rerandomizable RCCA-secure public key encryption and its applications. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 159–190. Springer, Heidelberg, Dec. 2019.
25. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.
26. M. Green. Secure blind decryption. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 265–282. Springer, Heidelberg, Mar. 2011.
27. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, Dec. 2006.
28. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, Apr. 2008.
29. A. Guillevic. `https://members.loria.fr/AGuillevic/pairing-friendly-curves/`.
30. S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu. TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF. In D. Gollmann, A. Miyaji, and H. Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 39–58. Springer, Heidelberg, July 2017.
31. C. S. Jutla and A. Roy. Improved structure preserving signatures under standard bilinear assumptions. In S. Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 183–209. Springer, Heidelberg, Mar. 2017.
32. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, Mar. 2011.
33. P. Kelley, S. Komanduri, M. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. Cranor, and J. Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. pages 523–537, 05 2012.
34. E. Kiltz and H. Wee. Quasi-adaptive NIZK for linear subspaces revisited. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, Apr. 2015.
35. H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, Aug. 2010.

36. B. Libert, T. Peters, and C. Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 247–276. Springer, Heidelberg, Mar. 2017.
37. P. D. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. *Journal of Cryptology*, 19(1):27–66, Jan. 2006.
38. M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 517–534. Springer, Heidelberg, Aug. 2007.
39. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, Nov. 1979.

A Signatures

This section first gives the definition of strong one-time security of signature schemes. It then presents Groth's one-time signature scheme as well as a structure-preserving existentially unforgeable signature scheme due to Jutla and Roy [31].

Strong One-Time Security. A signature scheme is one-time strongly unforgeable against chosen-message attacks if no efficient adversary can forge a signature on a message even after obtaining a single signature (potentially on the same message). That is to say, for all $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{Vf}(vk, M^*, \sigma^*) = 1 & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q_{\mathsf{Sign}} \leftarrow \emptyset \\ \wedge (M^*, \sigma^*) \notin Q_{\mathsf{Sign}} & (vk, sk) \leftarrow \mathsf{KG}(pp) \\ & (M^*, \sigma^*) \leftarrow \mathcal{A}^{O.\mathsf{Sign}^{\mathrm{OT}}(sk,\cdot)}(vk) \end{array}\right]$$

is negligible. Oracle $O.\mathsf{Sign}^{\mathrm{OT}}(sk, \cdot)$ can be queried only once, say on a message M. It computes and returns $\sigma \leftarrow \mathsf{Sign}(sk, M)$, and adds (M, σ) to Q_{Sign}.

A.1 Groth's Strong One-Time Signatures

Given a group generator G and family of hash functions $\mathcal{H}$, Groth's [27] signature scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $(p, \mathbb{G} = \langle g \rangle) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Generate $H \leftarrow_\$ \mathcal{H}$. Set and return $pp \leftarrow (p, \mathbb{G}, H)$.

$\mathsf{KG}(pp) \to (vk, sk)$**:** generate $x, y \leftarrow_\$ \mathbb{Z}_p^*$. Compute $f \leftarrow g^x$ and $h \leftarrow g^y$. Generate $r, s \leftarrow_\$ \mathbb{Z}_p^*$ and compute $c \leftarrow f^r h^s$. Set $vk \leftarrow (f, h, c)$ and $sk \leftarrow (x, y, r, s)$. Return (vk, sk).

$\mathsf{Sign}\left(sk, M \in \{0,1\}^*\right) \to \sigma$: generate $t \leftarrow_\$ \mathbb{Z}_p$, and compute and return $\sigma \leftarrow \left(t, (x(r-t) + ys - H(M))y^{-1}\right)$.

$\mathsf{Vf}(vk, M, \sigma) \to b \in \{0,1\}$: parse σ as (t, u). Return 1 if $c = g^{H(M)} f^t h^u$, else return 0.

Groth proved [27, Theorem 18] that it is one-time strongly unforgeable against chosen-message attacks if the discrete-logarithm assumption over G holds and if $\mathcal{H}$ is a family of collision-resistant hash functions.

A.2 Jutla and Roy's Signature Scheme

The following signature scheme, parametrized by a bilinear structure generator G, is due to Jutla and Roy [31]. It allows to sign vectors of first-group elements, and it is existentially unforgeable under the SXDH assumption.

$\mathsf{Setup}\left(1^\lambda, n\right) \to pp$: run $\Gamma \leftarrow (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Return $pp \leftarrow (\Gamma, n)$.

$\mathsf{KG}(pp) \to (vk, sk)$**:** Generate $b, d, f, k_0, A, z \leftarrow_\$ \mathbb{Z}_p$, $\mathbf{k} \leftarrow_\$ \mathbb{Z}_p^n$, $\mathbf{K} \leftarrow_\$ \mathbb{Z}_p^{n+4}$. Set vk[7] $\leftarrow \left(g_2^{AK_1}, \ldots, g_2^{AK_{n+4}}, g_2^{Az}, g_2^{A}\right)$ and $sk \leftarrow (b, d, f, k_0, \mathbf{k}, \mathbf{K}, z)$. Return (vk, sk).

$\mathsf{Sign}\left(sk, \mu \in \mathbb{G}_1^n\right) \to \sigma$: Generate $r, t \leftarrow_\$ \mathbb{Z}_p$. Compute

$$\rho \leftarrow g_1^r, \hat{\rho} \leftarrow g_1^{rb}, \psi \leftarrow g_1^{rt}, \gamma \leftarrow \prod_{i=1}^{n} \mu_i^{k_i} g_1^{k_0 + dr + frt}$$

$$\tau \leftarrow g_2^t, \pi \leftarrow \prod_{i=1}^{n} \mu_i^{K_i} \rho^{K_{n+1}} \hat{\rho}^{K_{n+2}} \psi^{K_{n+3}} \gamma^{K_{n+4}} g_1^z$$

Set and return $\sigma \leftarrow (\rho, \hat{\rho}, \psi, \gamma, \pi, \tau) \in \mathbb{G}_1^5 \times \mathbb{G}_2$.

$\mathsf{Vf}(vk, \mu, \sigma) \to b \in \{0,1\}$: If $e(\rho, \tau) = e(\psi, g_2)$ and

$$e\left(\pi, g_2^A\right) = \prod_{i=1}^{n} e\left(\mu_i, g_2^{AK_i}\right) e\left(\rho, g_2^{AK_{n+1}}\right) e\left(\hat{\rho}, g_2^{AK_{n+2}}\right) e\left(\psi, g_2^{AK_{n+3}}\right) e\left(\gamma, g_2^{AK_{n+4}}\right) e\left(g_1, g_2^{Az}\right)$$

then return 1, else return 0.

B Public-Key Encryption

This section recalls some security definitions for public-key encryption schemes as well as instantiations that are building blocks of the construction in Section 4.

B.1 Security Definitions

This section recalls the formal definitions of IND-PCA and IND-RCCA security.

IND-PCA Security. A labeled encryption scheme $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is IND-PCA secure if for every $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l| l} & pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q \leftarrow \emptyset; (pk, sk) \leftarrow \mathsf{KG}(pp) \\ & (st, \ell^*, M_0, M_1) \leftarrow \mathcal{A}^{O(Q, sk, \cdot)}(pk) \\ & b \leftarrow_\$ \{0,1\}; C^* \leftarrow \mathsf{Enc}^{\ell^*}(pk, M_b) \\ b = b' : & b' \leftarrow_\$ \mathcal{A}^{O(Q, sk, \cdot)}(st, C^*) \\ & \text{if } (\ell^*, C^*) \in Q \\ & \quad b' \leftarrow_\$ \{0,1\} \\ & \quad \text{return } (b, b') \\ & \text{return } (b, b') \end{array}\right]$$

[7] The verification key is independent of $b, d, f, k_0, \mathbf{k}$ as signatures are split-CRS quasi-adaptive proofs for an affine language defined by the message, b, d, f, k_0 and $\mathbf{k}$, and these proofs are simulated with $(\mathbf{K}, z)$ as trapdoor.

is negligibly close to $1/2$ (the *advantage* of $\mathcal{A}$ is then the distance of that probability to $1/2$). Oracle $\mathcal{O}$, with a state ($\mathcal{Q}$ and sk), replies to a (ℓ, C, M) query by returning the truth value of $\left(\mathsf{Dec}^{\ell}(sk, C) = M\right)$ and setting $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\ell, C)\}$. If $\mathcal{A}$ ever queries $\mathcal{O}$ on (ℓ^*, C^*), its advantage is set to 0 since b' is overwritten by a uniformly random bit.

IND-RCCA Security. An encryption scheme $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is IND-RCCA secure if for every $\lambda \in \mathbb{N}$, for every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} b = b' : \; pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); (pk, sk) \leftarrow \mathsf{KG}(pp) \\ \qquad\qquad (st, M_0, M_1) \leftarrow \mathcal{A}^{\mathcal{O}(sk, \perp, \perp, \cdot)}(pk) \\ \qquad\qquad b \leftarrow_{\$} \{0,1\}; C^* \leftarrow \mathsf{Enc}(pk, M_b) \\ \qquad\qquad b' \leftarrow_{\$} \mathcal{A}^{\mathcal{O}(sk, M_0, M_1, \cdot)}(st, C^*) \\ \qquad\qquad \text{return } (b, b') \end{array}\right]$$

is negligibly close to $1/2$. Oracle $\mathcal{O}$, with state (sk, M_0, M_1), replies to C queries by first computing $M \leftarrow \mathsf{Dec}(sk, C)$. If $M = M_0$ or $M = M_1$, it returns a special string replay indicating that C encrypts one of the challenge messages, otherwise it returns M.

B.2 Short Cramer–Shoup Encryption.

Given a *group generator* G (i.e., an algorithm which returns a prime p and the description of a p-order group on the input of a security parameter 1^{λ}) and a family $\mathcal{H}$ of hash functions, the (labeled) short Cramer–Shoup encryption scheme [1] consists of the following algorithms.

$\mathsf{Setup}\left(1^{\lambda}\right) \to pp$: generate $(p, \mathbb{G} = \langle g \rangle) \leftarrow \mathsf{G}\left(1^{\lambda}\right)$ and $H \leftarrow_{\$} \mathcal{H}$. Set and return $pp \leftarrow (p, \mathbb{G}, g, H)$.

$\mathsf{KG}(pp) \to (pk, sk)$: $sk \leftarrow (\zeta, \alpha, \beta, \alpha', \beta') \leftarrow_{\$} \mathbb{Z}_p^5$. Compute $pk \leftarrow (pp, h, \gamma, \delta) \leftarrow \left(pp, g^{\zeta},\ g^{\alpha} h^{\beta},\ g^{\alpha'} h^{\beta'}\right)$. (Parameters pp may further be omitted in the syntax.) Return (pk, sk).

$\mathsf{Enc}^{\ell}(pk, M \in \mathbb{G}; r \in \mathbb{Z}_p) \to C$: compute $U \leftarrow g^r$, $E \leftarrow h^r M$, $\xi \leftarrow H(U, E, \ell)$ and $V \leftarrow \left(\gamma \delta^{\xi}\right)^r$. Set and return $C \leftarrow (U, E, V)$.

$\mathsf{Dec}^{\ell}(sk, C) \to M/\perp$: compute $M \leftarrow U/E^{\zeta}$ and $\xi \leftarrow H(U, E, \ell)$. If $V = U^{\alpha + \xi \alpha'} (E/M)^{\beta + \xi \beta'}$ then return M, else return $\perp$.

Abdalla, Benhamouda and Pointcheval proved that this scheme is IND-PCA secure if the DDH assumption over G holds and if $\mathcal{H}$ is second-preimage resistant.

B.3 Encryption Scheme of Faonio, Fiore, Herranz and Ràfols

Faonio, Fiore, Herranz and Ràfols [24] constructed a publicly verifiable, re-randomizable, structure-preserving encryption scheme which is secure under the matrix decisional Diffie–Hellman assumption [23]. The explicit version of their scheme under the SXDH assumption is given below.

$\underline{\mathsf{Setup}\left(1^\lambda\right) \to pp:}$ run $\Gamma \leftarrow (p,\ \mathbb{G}_1 = \langle g_1 \rangle,\ \mathbb{G}_2 = \langle g_2 \rangle,\ \mathbb{G}_T,\ e) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Generate a Groth–Sahai common reference string, i.e., $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$ and $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$. Set and return $pp \leftarrow (\Gamma, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{w})$.

$\underline{\mathsf{KG}(pp) \to (pk, sk):}$ generate $D, E \leftarrow_\$ \mathbb{Z}_p$, $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 \leftarrow_\$ \mathbb{Z}_p$, $\mathbf{F} \leftarrow_\$ \mathbb{Z}_p^2$ and $\mathbf{G} \leftarrow_\$ \mathbb{Z}_p^{2\times 3}$. Compute

$$\mathbf{M}_1 \leftarrow \begin{bmatrix} g_1^D & & & & & \\ g_1^{\beta_1 D+\beta_2} & a_1\ b_1 & & & & \\ 1_{\mathbb{G}_1} & a_2\ b_2 & & & & \\ g_1^{F_{1,1}D+F_{2,1}} & & a_1\ b_1 & & & \\ 1_{\mathbb{G}_1} & & a_2\ b_2 & & & \\ g_1^{F_{1,2}D+F_{2,2}} & & & & a_1\ b_1 & \\ 1_{\mathbb{G}_1} & & & & a_2\ b_2 & \end{bmatrix}$$

and

$$\mathbf{M}_2 \leftarrow \begin{bmatrix} g_2^E & & & & & & \\ g_2^{\gamma_1 D+\gamma_2} & v_1\ w_1 & & & & & \\ 1_{\mathbb{G}_2} & v_2\ w_2 & & & & & \\ g_2^{G_{1,1}E+G_{2,1}} & & v_1\ w_1 & & & & \\ 1_{\mathbb{G}_2} & & v_2\ w_2 & & & & \\ g_2^{G_{1,2}E+G_{2,2}} & & & & v_1\ w_1 & & \\ 1_{\mathbb{G}_2} & & & & v_2\ w_2 & & \\ g_2^{G_{1,3}E+G_{2,3}} & & & & & v_1\ w_1 & \\ 1_{\mathbb{G}_2} & & & & & v_2\ w_2 & \end{bmatrix}.$$

For $i = 1, 2$, generate $A_i \leftarrow_\$ \mathbb{Z}_p$. Generate $\mathbf{K}_i \leftarrow_\$ \mathbb{Z}_p^{(5+2i)\times 1}$. Compute matrices $\mathbf{P}_i \leftarrow \mathbf{M}_i^{\mathrm{T}}\mathbf{K}_i$ and $\mathbf{C}_i \leftarrow A_i\mathbf{K}_i$. Set

$$\begin{aligned} pk \leftarrow \Bigg(&\Gamma, g_1^D, g_1^{\alpha_1 D+\alpha_2}, g_1^{\beta_1 D+\beta_2}, g_2^E, g_2^{\gamma_1 E+\gamma_2}, \\ &\begin{bmatrix} g_1^{F_{1,1}D+F_{2,1}} \\ g_1^{F_{1,2}D+F_{2,2}} \end{bmatrix}, \begin{bmatrix} g_1^{G_{1,1}D+G_{1,2}+G_{1,3}(\alpha_1 D+\alpha_2)} \\ g_1^{G_{2,1}D+G_{2,2}+G_{2,3}(\alpha_1 D+\alpha_2)} \end{bmatrix}, \\ &\begin{bmatrix} g_2^{G_{1,1}E+G_{2,1}} \\ g_2^{G_{1,2}E+G_{2,2}} \\ g_2^{G_{1,3}E+G_{2,3}} \end{bmatrix}, \begin{bmatrix} g_2^{F_{1,1}E+F_{1,2}} \\ g_2^{F_{2,1}E+F_{2,2}} \end{bmatrix}, A_1, A_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{C}_1, \mathbf{C}_2 \Bigg) \\ sk \leftarrow (&\Gamma, \alpha_1, \alpha_2, A_1, A_2, \mathbf{C}_1, \mathbf{C}_2)\,. \end{aligned}$$

Return (pk, sk). If ciphetext blinding is necessary as in Section 4.1, store also $\mathbf{G}$ in sk.

Matrices $\mathbf{M}_i$, $\mathbf{P}_i$ and $\mathbf{C}_i$, and scalars A_i are parameters for the quasi-adaptive non-interactive zero-knowledge proof system of Kiltz and Wee for linear spaces [34] with witness-sampleable distribution.

$\underline{\mathsf{Enc}(pk, M \in \mathbb{G}_1) \to C :}$

- compute a tag, i.e.,
  - $r, s \leftarrow_\$ \mathbb{Z}_p$
  - $\mathbf{x}^{\mathrm{T}} \leftarrow \left[g_1^{rD} \; g_1^{r} \; g_1^{r(\alpha_1 D + \alpha_2)} M \right]$
  - $\mathbf{y}^{\mathrm{T}} \leftarrow \left[g_2^{sE} \; g_2^{s} \right]$
- compute a smooth projective hash π of the tag, i.e.,

$$\pi \leftarrow e\left(g_1^{r(\beta_1 D + \beta_2)}, g_2\right) \prod_{i=1,2} e\left(g_1^{r\left(F_{1,i} D + F_{2,i}\right)}, y_i\right) e\left(g_1, g_2^{s(\gamma_1 E + \gamma_2)}\right) \prod_{i=1,2,3} e\left(x_i, g_2^{s\left(G_{1,i} E + G_{2,i}\right)}\right)$$

- prove that π is well-formed, i.e.,
  - $r_{av}, r_{bv}, r_{aw}, r_{bw}, \rho_1, \rho_2, \sigma_{i,j}, r_i, s_{k,j} \leftarrow_\$ \mathbb{Z}_p$ for $i, j \in \{1, 2\}$ and $k = 1, 2, 3$
  - $\mathbf{c}(\pi) \leftarrow \left[\pi_{i,j}\right]_{i,j=1,2} f\left(\mathbf{a}^{r_{av}}, \mathbf{v}\right) f\left(\mathbf{b}^{r_{bv}}, \mathbf{v}\right) f\left(\mathbf{a}^{r_{aw}}, \mathbf{w}\right) f\left(\mathbf{b}^{r_{bw}}, \mathbf{w}\right)$, with $\pi_{i,j} \coloneqq \pi$ if $i = j = 2$ and $1_{\mathbb{G}_T}$ otherwise
  - $\mathbf{c}_0 \leftarrow \left[1 \; g_1^{r(\beta_1 D + \beta_2)}\right]^{\mathrm{T}} \mathbf{a}^{\rho_1} \mathbf{b}^{\rho_2}$
  - $\mathbf{c}_{1,i} \leftarrow \left[1 \; g_1^{r\left(F_{1,i} D + F_{2,i}\right)}\right]^{\mathrm{T}} \mathbf{a}^{\sigma_{i,1}} \mathbf{b}^{\sigma_{i,2}}$ for $i = 1, 2$
  - $\mathbf{d}_0 \leftarrow \left[1 \; g_2^{s(\gamma_1 E + \gamma_2)}\right]^{\mathrm{T}} \mathbf{v}^{r_1} \mathbf{w}^{r_2}$
  - $\mathbf{d}_{1,i} \leftarrow \left[1 \; g_2^{s\left(G_{1,i} E + G_{2,i}\right)}\right]^{\mathrm{T}} \mathbf{v}^{s_{i,1}} \mathbf{w}^{s_{i,2}}$ for $i = 1, 2, 3$
  - $\mathbf{T} \leftarrow_\$ \mathbb{Z}_p^{2\times 2}$
  - $\Theta \leftarrow \begin{bmatrix} a_1^{T_{1,1}} b_1^{T_{2,1}} & a_2^{T_{1,1}} b_2^{T_{1,2}} g_1^{r_1} x_1^{s_{1,1}} x_2^{s_{2,1}} x_3^{s_{3,1}} \\ a_1^{T_{1,2}} b_1^{T_{2,2}} & a_2^{T_{2,1}} b_2^{T_{2,2}} g_1^{r_2} x_1^{s_{1,2}} x_2^{s_{2,2}} x_3^{s_{3,2}} \end{bmatrix}^{\mathrm{T}} =: \left[\Theta_1 \; \Theta_2\right]$
  - $\tilde{\Pi} \leftarrow \begin{bmatrix} v_1^{-T_{1,1}} w_1^{-T_{2,1}} & v_2^{-T_{1,1}} w_2^{-T_{2,1}} g_2^{\rho_1} y_1^{\sigma_{1,1}} y_2^{\sigma_{2,1}} \\ v_1^{-T_{1,2}} w_1^{-T_{2,2}} & v_2^{-T_{1,2}} w_2^{-T_{2,2}} g_2^{\rho_2} y_1^{\sigma_{1,2}} y_2^{\sigma_{2,2}} \end{bmatrix}$
  - $\Pi_1 \leftarrow \tilde{\Pi}_1^{\mathrm{T}} \mathbf{v}^{r_{av}} \mathbf{w}^{r_{aw}}; \Pi_2 \leftarrow \tilde{\Pi}_2^{\mathrm{T}} \mathbf{v}^{r_{bv}} \mathbf{w}^{r_{bw}}; \Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix}$
- prove that the commitments are well-formed, i.e., that $\left[g_1^{Dr} \; \mathbf{c}_0^{\mathrm{T}} \; \mathbf{c}_{1,1}^{\mathrm{T}} \; \mathbf{c}_{1,2}^{\mathrm{T}}\right]^{\mathrm{T}}$ is in span $(\mathbf{M}_1)$ and that $\left[g_2^{Es} \; \mathbf{d}_0^{\mathrm{T}} \; \mathbf{d}_{1,2}^{\mathrm{T}} \; \mathbf{d}_{1,2}^{\mathrm{T}} \; \mathbf{d}_{1,3}^{\mathrm{T}}\right]^{\mathrm{T}}$ is in span $(\mathbf{M}_2)$. That is, for $j = 1, 2$, compute

$$\Psi_1 \leftarrow g_1^{rP_{1,1} + \sum_{i=1,2} \rho_i P_{1,1+i} + \sum_{i,j=1,2} \sigma_{i,j} P_{1,2i+j+1}}$$
$$\Psi_2 \leftarrow g_2^{sP_{2,1} + \sum_{i=1,2} r_i P_{2,1+i} + \sum_{i=1}^{3} \sum_{j=1}^{2} s_{i,j} P_{2,2i+j+1}}$$

- set and return

$$C \leftarrow \left(\mathbf{x}, \mathbf{y}, \mathbf{c}(\pi), \mathbf{c}_0, (\mathbf{c}_{1,i})_{i=1}^2, \mathbf{d}_0, (\mathbf{d}_{1,i})_{i=1}^3, \Pi, \Theta, (\Psi_i)_{i=1,2}\right).$$

A ciphertext comprises 14 $\mathbb{G}_1$ elements, 15 $\mathbb{G}_2$ elements and 4 $\mathbb{G}_T$ elements.

$\underline{\mathsf{Dec}(sk, C) \to M/\perp :}$

- verify that the commitments are well-formed, i.e., that

$$e\left(\Psi_1, g_2^{A_1}\right) = e\left(g_1^{Dr}, g_2^{\mathbf{C}_{1,1}}\right) \prod_{i=1,2} e\left(\mathbf{c}_{0,i}, g_2^{\mathbf{C}_{1,1+i}}\right) \prod_{i,j=1,2} e\left(\mathbf{c}_{1,i,j}, g_2^{\mathbf{C}_{1,2i+j+1}}\right)$$

  and that

$$e\left(g_1^{A_2}, \Psi_2\right) = e\left(g_1^{\mathbf{C}_{2,1}}, g_2^{Es}\right) \prod_{i=1,2} e\left(g_1^{\mathbf{C}_{2,1+i}}, \mathbf{d}_{0,i}\right) \prod_{i=1,2,3} \prod_{j=1,2} e\left(g_1^{\mathbf{C}_{2,2i+j+1}}, \mathbf{d}_{1,i,j}\right)$$

- verify that the opening to $\mathbf{c}(\pi)$ is the smooth projective hash of the tag, i.e., that

$$f\left(\mathbf{c}_0, \begin{bmatrix} 1 \\ g_2 \end{bmatrix}\right) \prod_{i=1,2} f\left(\mathbf{c}_{1,i}, \begin{bmatrix} 1 \\ y_i \end{bmatrix}\right)$$
$$f\left(\begin{bmatrix} 1 \\ g_1 \end{bmatrix}, \mathbf{d}_0\right) \prod_{i=1,2,3} f\left(\begin{bmatrix} 1 \\ x_i \end{bmatrix}, \mathbf{d}_{1,i}\right)$$
$$= f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right) f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_2, \mathbf{w}\right) \mathbf{c}(\pi)$$

- if both verifications succeed, compute and return $M \leftarrow x_3 x_1^{-\alpha_1} x_2^{-\alpha_2}$, else return $\perp$.

$\underline{\mathsf{Vf}(pk, C) \to b :}$ do the same verifications as algorithm Dec (they do not require knowledge of sk, only of pk). If they succeed, return 1, else return 0.

$\underline{\mathsf{Rand}(pk, C) \to \hat{C} :}$ first verify that the ciphertext is valid. Next, re-randomize the tag, and all proofs and commitments, i.e., do

- $\hat{r}, \hat{s} \leftarrow_{\$} \mathbb{Z}_p$
- $\hat{\mathbf{x}} \leftarrow \mathbf{x} \left[g_1^{\hat{r}D} \; g_1^{\hat{r}} \; g_1^{\hat{r}(\alpha_1 D + \alpha_2)}\right]^{\mathrm{T}}$
- $\hat{\mathbf{y}} \leftarrow \mathbf{y} \left[g_2^{\hat{s}E} g_2^{\hat{s}}\right]^{\mathrm{T}}$
- re-randomize π, i.e., compute

$$\hat{\pi} \leftarrow \pi \cdot e\left(g_1^{\hat{r}(\beta_1 D + \beta_2)}, g_2\right) \prod_{i=1,2} e\left(g_1^{\hat{r}\left(F_{1,i} D + F_{2,i}\right)}, y_i\right) e\left(g_1, g_2^{\hat{s}(\gamma_1 E + \gamma_2)}\right) \prod_{i=1,2,3} e\left(x_i, g_2^{\hat{s}\left(G_{1,i} E + G_{2,i}\right)}\right)$$

– $\hat{r}_{av}, \hat{r}_{bv}, \hat{r}_{aw}, \hat{r}_{bw}, \hat{\rho}_1, \hat{\rho}_2, \hat{\sigma}_{i,j}, \hat{r}_i, \hat{s}_{k,j} \leftarrow_\$ \mathbb{Z}_p$ for $i, j \in \{1,2\}$ and $k = 1,2,3$
– $\hat{\mathbf{c}}(\pi) \leftarrow f\left(\mathbf{a}^{\hat{r}_{av}}, \mathbf{v}\right) f\left(\mathbf{b}^{\hat{r}_{bv}}, \mathbf{v}\right) f\left(\mathbf{a}^{\hat{r}_{aw}}, \mathbf{w}\right) f\left(\mathbf{b}^{\hat{r}_{bw}}, \mathbf{w}\right) \mathbf{c}(\pi)$
– $\hat{\mathbf{c}}_0 \leftarrow \mathbf{c}_0 \mathbf{a}^{\hat{\rho}_1} \mathbf{b}^{\hat{\rho}_2}$
– $\hat{\mathbf{c}}_{1,i} \leftarrow \mathbf{c}_{1,i} \mathbf{a}^{\hat{\sigma}_{i,1}} \mathbf{b}^{\hat{\sigma}_{i,2}}$ for $i = 1,2$
– $\hat{\mathbf{d}}_0 \leftarrow \mathbf{d}_0 \mathbf{v}^{\hat{r}_1} \mathbf{w}^{\hat{r}_2}$
– $\hat{\mathbf{d}}_{1,i} \leftarrow \mathbf{d}_{1,i} \mathbf{v}^{\hat{s}_{i,1}} \mathbf{w}^{\hat{s}_{i,2}}$ for $i = 1,2,3$
– $\hat{\mathbf{T}} \leftarrow_\$ \mathbb{Z}_p^{2\times 2}$
– $\hat{\Theta} \leftarrow \Theta \circ \begin{bmatrix} a_1^{\hat{T}_{1,1}} b_1^{\hat{T}_{2,1}} & a_2^{\hat{T}_{1,1}} b_2^{\hat{T}_{1,2}} g_1^{\hat{r}_1} x_1^{\hat{s}_{1,1}} x_2^{\hat{s}_{2,1}} x_3^{\hat{s}_{3,1}} \\ a_1^{\hat{T}_{1,2}} b_1^{\hat{T}_{2,2}} & a_2^{\hat{T}_{2,1}} b_2^{\hat{T}_{2,2}} g_1^{\hat{r}_2} x_1^{\hat{s}_{1,2}} x_2^{\hat{s}_{2,2}} x_3^{\hat{s}_{3,2}} \end{bmatrix}^{\mathrm{T}}$
– compute

$$\hat{\Pi} \leftarrow \Pi \circ \begin{bmatrix} v_1^{-\hat{T}_{1,1}} w_1^{-\hat{T}_{2,1}} & v_2^{-\hat{T}_{1,1}} w_2^{-\hat{T}_{2,1}} g_2^{\hat{\rho}_1} y_1^{\hat{\sigma}_{1,1}} y_2^{\hat{\sigma}_{2,1}} \\ v_1^{-\hat{T}_{1,2}} w_1^{-\hat{T}_{2,2}} & v_2^{-\hat{T}_{1,2}} w_2^{-\hat{T}_{2,2}} g_2^{\hat{\rho}_2} y_1^{\hat{\sigma}_{1,2}} y_2^{\hat{\sigma}_{2,2}} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} v^{\hat{r}_{av}} w^{\hat{r}_{aw}} \\ v^{\hat{r}_{bv}} w^{\hat{r}_{bw}} \end{bmatrix}$$

– $\hat{\Psi}_1 \leftarrow \Psi_1 \cdot g_1^{\hat{r} P_{1,1} + \sum_{i=1,2} \hat{\rho}_i P_{1,1+i} + \sum_{i,j=1,2} \hat{\sigma}_{i,j} P_{1,2i+j+1}}$
– $\hat{\Psi}_2 \leftarrow \Psi_2 \cdot g_2^{\hat{s} P_{2,1} + \sum_{i=1,2} \hat{r}_i P_{2,1+i} + \sum_{i=1}^{3} \sum_{j=1}^{2} \hat{s}_{i,j} P_{2,2i+j+1}}$
– set and return $\hat{C} \leftarrow \left(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{c}}(\pi),\ \hat{\mathbf{c}}_0,\ \left(\hat{\mathbf{c}}_{1,i}\right)_{i=1}^{2},\ \hat{\mathbf{d}}_0,\ \left(\hat{\mathbf{d}}_{1,i}\right)_{i=1}^{3},\ \hat{\Pi},\ \hat{\Theta},\ \left(\hat{\Psi}_{i,j}\right)_{i,j=1}^{2}\right)$.

Threshold Decryption. The scheme of Faonio et al. can be turned into a t-out-of-n scheme by doing a t-out-of-n Shamir share [39] of α_1 and α_2. Before partially decrypting the plaintext, each shareholder first verifies the validity of the ciphertext, which is possible as the scheme is publicly verifiable. If the ciphertext is invalid, the shareholder returns $\perp$. As a result, any $t+1$ shares are sufficient to decrypt ciphertexts, but no information about the plaintexts can be inferred with only t shares.

Verification of Blinded Ciphertexts. This section gives an algorithm to blind the plaintext underlying a ciphertext of the scheme of Faonio et al. without the knowledge of the secret key, and an algorithm to verify that the blinding was done correctly given a share of the secret key and auxiliary information provided by the first algorithm.

Formally, these algorithm are as follows.

$\mathsf{Blind}(pk, C) \rightarrow \left(\tilde{C}, aux\right)$:

∘ if $\mathsf{Vf}(pk, C) = 0$ return $\perp$
∘ parse $C = \left(\mathbf{x},\ \mathbf{y},\ \mathbf{c}(\pi),\ \mathbf{c}_0,\ (\mathbf{c}_{1,i})_{i=1}^{2},\ \mathbf{d}_0,\ (\mathbf{d}_{1,i})_{i=1}^{3},\ \Pi,\ \Theta,\ (\Psi_i)_{i=1,2}\right)$
∘ generate a blinding factor $R \leftarrow_\$ \mathbb{G}_1$

- blind the payload part of the ciphertext, i.e., $\tilde{x}_3 \leftarrow x_3 R$
- $\tilde{\mathbf{x}} \leftarrow \left[x_1\ x_2\ \tilde{x}_3\right]^{\mathrm{T}}$
- $\tilde{\pi} \leftarrow \pi e\left(R, g_2^{s(G_{1,3}E+G_{2,3})}\right)$
- denote by $\mathbf{c}(\tilde{\pi})$ the matrix $\mathbf{c}(\pi)\left[\tilde{\pi}_{i,j}\right]_{i,j=1,2}$, with $\tilde{\pi}_{i,j} \coloneqq \tilde{\pi}$ if $i = j = 2$ and $1_{\mathbb{G}_T}$ otherwise. It is a commitment to $\tilde{\pi}$ with the same randomness as for $\mathbf{c}(\pi)$
- set $\tilde{C} \leftarrow (\tilde{\mathbf{x}},\ \mathbf{y},\ \mathbf{c}(\pi),\ \mathbf{c}_0,\ (\mathbf{c}_{1,i})_{i=1}^2,\ \mathbf{d}_0,\ (\mathbf{d}_{1,i})_{i=1}^3,\ \Pi,\ \Theta,\ (\Psi_i)_{i=1,2})$. Note that $\mathbf{c}(\pi)$ is *not* replaced by $\mathbf{c}(\tilde{\pi})$, as one would not be able to prove that it is well-formed for the scheme of Faonio et al. is RCCA-secure
- commit to R, i.e.,
  - $\sigma_1, \sigma_2 \leftarrow_{\$} \mathbb{Z}_p$; $\mathbf{c}(R) \leftarrow \left[1\ R\right]^{\mathrm{T}} \mathbf{a}^{\sigma_1} \mathbf{b}^{\sigma_2}$
- Note that

$$\mathbf{c}(\pi) f\left(\mathbf{c}(R), \begin{bmatrix} 1 \\ g_2^{s(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) = f\left(\mathbf{a}, \begin{bmatrix} 1 \\ g_2^{s\sigma_1(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) f\left(\mathbf{b}, \begin{bmatrix} 1 \\ g_2^{s\sigma_2(G_{1,3}E+G_{2,3})} \end{bmatrix}\right) \mathbf{c}(\tilde{\pi}).$$

  The issue is that without the secret key, one cannot compute $g_2^{s\sigma_i(G_{1,3}E+G_{2,3})}$ for $i = 1, 2$. However, g_2^s and g_2^{sE} are part of the ciphertext, so the user can compute

$$\Sigma \leftarrow \begin{bmatrix} g_2^{s\sigma_1} & g_2^{s\sigma_2} \\ g_2^{s\sigma_1 E} & g_2^{s\sigma_2 E} \end{bmatrix}.$$

  Given Σ and $\mathbf{G}$ (which is part of the secret key), one can then compute $g_2^{s\sigma_1(G_{1,3}E+G_{2,3})}$ and $g_2^{s\sigma_1(G_{1,3}E+G_{2,3})}$
- $aux \leftarrow (\mathbf{c}(R), \Sigma)$. It is essentially a designated GS proof intended for any party who knows $\mathbf{G}$
- return $\left(\tilde{C}, aux\right)$.

$\mathsf{BlindVf}\left(sk, \tilde{C}, aux\right) \rightarrow b \in \{0,1\}$:

- verify that commitments $\mathbf{c}_0, (\mathbf{c}_{1,i})_{i=1}^2, \mathbf{d}_0, (\mathbf{d}_{1,i})_{i=1}^3$, are well-formed as in algorithm Dec
- compute $\Sigma_1 \leftarrow \Sigma_{1,1}^{G_{2,3}} \Sigma_{2,1}^{G_{1,3}}$ and $\Sigma_2 \leftarrow \Sigma_{1,2}^{G_{2,3}} \Sigma_{2,2}^{G_{1,3}}$. This requires the matrix $\mathbf{G}$ in the secret key. If this matrix is kept in a share of the secret key, then the key share is enough.

- verify with the corrective terms Σ_1 and Σ_2 that the opening to $\mathbf{c}(\pi)$ is the smooth projective hash of the tag, i.e., that

$$\begin{aligned}
&f\left(\mathbf{c}_0, \begin{bmatrix}1\\ g_2\end{bmatrix}\right) \prod_{i=1,2} f\left(\mathbf{c}_{1,i}, \begin{bmatrix}1\\ y_i\end{bmatrix}\right) f\left(\begin{bmatrix}1\\ g_1\end{bmatrix}, \mathbf{d}_0\right) \prod_{i=1,2,3} f\left(\begin{bmatrix}1\\ \tilde{x}_i\end{bmatrix}, \mathbf{d}_{1,i}\right) \\
=&f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right) f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_2, \mathbf{w}\right) \mathbf{c}(\pi) f\left(\mathbf{c}(R), \begin{bmatrix}1\\ g_2^{s\left(G_{1,3}E+G_{2,3}\right)}\end{bmatrix}\right) \\
&f\left(\mathbf{a}, \begin{bmatrix}1\\ \Sigma_1^{-1}\end{bmatrix}\right) f\left(\mathbf{b}, \begin{bmatrix}1\\ \Sigma_2^{-1}\end{bmatrix}\right)
\end{aligned}$$

- if both verifications succeed, return 1, else return 0.

C Smooth Projective Hash Functions

This section explains how smooth projective hash function can be viewed as proofs of membership with designated verifier, and recalls a KV-SPHF for short Cramer–Shoup ciphertexts.

C.1 Designated-Verifier Proofs of Membership

SPHFs can be seen as designated-verifier proofs of membership to $\mathcal{L}$ [2, 12]. Indeed, to prove that a word C is in $\mathcal{L}$, a designated verifier can generate a key hk and compute a projective key hp which she sends to the prover. Given a membership witness w, the prover evaluates the SPHF on (C, w) with hp and sends the result as a proof to the verifier. To verify the proof, the verifier computes the SPHF on C using hk and accepts if and only if the result matches the proof value. The correctness of the protocol follows from the correctness of the SPHF, and its soundness from the adaptive smoothness of the SPHF.

C.2 KV-SPHF for Short Cramer–Shoup Ciphertexts

For a fixed tuple $(p, \mathbb{G} = \langle g \rangle)$, denote by $\mathcal{L}_M^\ell$ the language $\left\{C\colon \exists r,\ \ C = (U, E, V) = \left(g^r, h^r M, \left(\gamma\delta^{H(g^r, h^r M, \ell)}\right)^r\right)\right\}$. Given a group generator G, $\Gamma \leftarrow \mathsf{G}(p, \mathbb{G} = \langle g \rangle)$ and $M \in G$, the following scheme, due to Abdalla, Benhamouda and Pointcheval [1], is a perfectly smooth KV-SPHF for $\mathcal{L}_M^\ell$.

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: generate $sk \leftarrow (\zeta, \alpha, \beta, \alpha', \beta') \leftarrow_\$ \mathbb{Z}_p^5$. Compute $pk \leftarrow (h, \gamma, \delta) \leftarrow \left(g^\zeta,\ g^\alpha h^\beta,\ g^{\alpha'} h^{\beta'}\right)$. Return (Γ, pk, sk).

$\mathsf{HashKG}\left(\mathcal{L}_M^\ell\right) \to hk$: return $hk \leftarrow (\lambda, \mu, \nu, \theta) \leftarrow_\$ \mathbb{Z}_p^4$.

$\mathsf{ProjKG}\left(hk, \mathcal{L}_M^\ell, \bot\right) \to hp$: return $hp \leftarrow \left(g^\lambda h^\nu \gamma^\theta,\ g^\mu \delta^\theta\right)$. Note that hp actually depends on neither M nor ℓ.

$\mathsf{Hash}\left(hk, \mathcal{L}_M^\ell, C\right) \to \mathfrak{H}$: compute $\xi \leftarrow H(U, E, \ell)$. Return $\mathfrak{H} \leftarrow U^{\lambda+\mu\xi} (E/M)^\nu V^\theta$.

$\mathsf{ProjHash}\left(hp, \mathcal{L}_M^\ell, C, r\right) \to \mathfrak{H}$: compute $\xi \leftarrow H(U, E, \ell)$. Return $\mathfrak{H} \leftarrow \left(hp_1 hp_2^\xi\right)^r$.

D Proof Systems

A proof system $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf})$ is *correct* if for all $\lambda \in \mathbb{N}$, for all $crs \leftarrow \mathsf{Setup}(1^\lambda)$, for all $(x, w) \in \mathcal{R}$, $\Pr[\mathsf{Vf}(crs, x, \mathsf{Prove}(crs, x, w)) = 1] = 1$.

Π satisfies $\varepsilon^{\text{sound}}$-*soundness* if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{Vf}(crs, x, \pi) = 1 \wedge x \notin \mathcal{L} : \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(crs) \end{array}\right] \leq \varepsilon^{\text{sound}}.$$

Π is ε^{wi}-*witness-indistinguishable* if for all $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$,

$$\left|\Pr\left[b = b' : \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (st, x, w_0, w_1) \leftarrow \mathcal{A}(crs) \\ b \leftarrow_{\$} \{0,1\} \\ \text{if } (x, w_0) \notin \mathcal{R} \text{ or } (x, w_1) \notin \mathcal{R} \\ \quad b' \leftarrow_{\$} \{0,1\} \\ \quad \text{return } (b, b') \\ \pi \leftarrow \mathsf{Prove}(crs, x, w_b) \\ b' \leftarrow \mathcal{A}(st, \pi) \\ \text{return } (b, b') \end{array}\right] - 1/2\right| \leq \varepsilon^{\text{wi}}.$$

Π is $(q, \varepsilon^{\text{zk}})$-*zero-knowledge* if there exist two algorithms $\mathsf{TSetup}(1^\lambda) \to (crs, \tau)$ and $\mathsf{Sim}(crs, \tau, x) \to \pi$ such that for all $\lambda \in \mathbb{N}$, the distribution of crs is the same as that of Setup, and such that for every PPT adversary $\mathcal{A}$ that makes at most q oracle queries,

$$\left|\Pr\left[b = 1 : \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ b \leftarrow \mathcal{A}^{O_{\mathsf{Prove}}(crs, \cdot)}(crs) \end{array}\right] - \Pr\left[b = 1 : \begin{array}{l} (crs, \tau) \leftarrow \mathsf{TSetup}(1^\lambda) \\ b \leftarrow \mathcal{A}^{O_{\mathsf{Sim}}(crs, \tau, \cdot)}(crs) \end{array}\right]\right| \leq \varepsilon^{\text{zk}},$$

with $O_{\mathsf{Prove}}(crs, \cdot)$ an oracle that computes and returns $\mathsf{Prove}(crs, x, w)$ on input $(x, w) \in \mathcal{R}$, and returns $\perp$ on input $(x, w) \notin \mathcal{R}$; and O_{Sim} an oracle that computes and returns $\mathsf{Sim}(crs, x, \tau)$ on input $(x, w) \in \mathcal{R}$, and returns $\perp$ on input $(x, w) \notin \mathcal{R}$.

D.1 Simulation Soundness under Controlled Malleability.

In certain cases, it is necessary to be able to simulate proofs while still ensuring that no PPT algorithm can compute new valid proofs for false statements without a trapdoor. This property is commonly known as simulation soundness. However, a proof system cannot a priori be both malleable and simulation sound as malleability allows to compute proofs on transformed words without the knowledge of a witness. Chase et al. [19] put forth a relaxed version of simulation soundness which allows for malleability while guaranteeing a meaningful form of soundness; namely an adversary cannot compute (without trapdoor) a valid proof for the membership of x to $\mathcal{L}$ if $x \notin \mathcal{L}$ and x is not the image under an admissible transformation (from a pre-determined class) of a word for which it was given a simulated proof. Their definition requires the proof system to be a proof of knowledge (i.e., it requires extractability), but the following definition is rather for proof of statements.

Formal Definition. Let $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{SS\_Setup}, \mathsf{Sim}, \mathsf{Eval})$ be a non-interactive zero-knowledge proof system for a language $\mathcal{L}$ which is malleable w.r.t. a class $\mathcal{T}$ of allowable transformations for the relation $\mathcal{R}$ relative to $\mathcal{L}$. The proof system is said to satisfy *Controlled-Malleable (CM) simulation soundness* w.r.t. $\mathcal{T}$ if for all $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|ll} \mathsf{Vf}(crs, x, \pi) = 1 & & (crs, \tau) \leftarrow \mathsf{SS\_Setup} \\ \wedge (x, \pi) \notin Q \wedge x \notin \mathcal{L} & : & Q \leftarrow \emptyset \\ \wedge \forall (T, x', *) \in \mathcal{T} \times Q, & & (x, \pi) \leftarrow \mathcal{A}^{O.\mathsf{Sim}(Q, crs, \tau, \cdot)}(crs) \\ \quad x \neq T_x(x') & & \end{array}\right]$$

is negligible, with $O.\mathsf{Sim}$ an oracle which computes simulated proofs π on inputs x and adds (x, π) to Q.

Generic Construction. Let $\mathcal{R}$ be an efficient relation with corresponding language $\mathcal{L}$. Consider a class $\mathcal{T}$ of allowable transformations for $\mathcal{R}$ containing the identity, and for which membership can be efficiently tested. Consider an existentially unforgeable signature scheme $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sign}, \mathsf{Vf})$, and a sound witness-indistinguishable proof system $\hat{\Pi} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf}, \mathsf{Eval})$ for the language:

$$\begin{aligned} \{(x \ := (x_1, x_2), vk) : \exists \left(w, x_1', T, \sigma\right), (x, w) \in \mathcal{R} \vee \\ \mathsf{SIG.Vf}\left(vk, x_1', \sigma\right) = 1 \wedge x = T_x\left(x_1', x_2\right) \wedge T \in \mathcal{T} \} . \end{aligned}$$

Suppose that it is partially extractable in the sense that there exists a trapdoor-setup algorithm which returns a trapdoor and a CRS indistinguishable from the output of Setup; and an extraction algorithm such that no PPT adversary has significant probability of computing a valid proof π on a word (x, vk), where the value $(*, x_1', *, \sigma)$ returned by the extractor on the input of the CRS, the trapdoor and (x, π) is such that $\mathsf{SIG.Vf}(vk, x_1', \sigma) = 0$. Let $\hat{\mathcal{T}}$ be a class of transformations such that for all $T' \in \mathcal{T}$, there exists $\hat{T} \in \hat{\mathcal{T}}$ such that $\hat{T}_x \colon (x, vk) \mapsto (T_x'(x), vk)$ and $\hat{T}_w \colon (w, x_1', T, \sigma) \mapsto \left(T_w'(w), x_1', T' \circ T, \sigma\right)$.

We devise a new proof system Π, described hereafter, which is inspired by the scheme of Chase et al. [19, Section 3]:

$\mathsf{Setup}\left(1^\lambda\right) \to crs$: run $crs' \leftarrow \hat{\Pi}.\mathsf{Setup}\left(1^\lambda\right)$, $pp \leftarrow \mathsf{SIG.Setup}\left(1^\lambda\right)$ and $(vk, sk) \leftarrow \mathsf{SIG.KG}(pp)$. Set and return $crs \leftarrow (crs', vk)$.

$\mathsf{Prove}(crs, x, w) \to \pi$: run $\pi \leftarrow \hat{\Pi}.\mathsf{Prove}\left(crs', (x, vk), \ (w, \perp)\right)$. Return π.

$\mathsf{Vf}(crs, x, \pi) \to b \in \{0, 1\}$: return $\hat{\Pi}.\mathsf{Vf}\left(crs', (x, vk), \pi\right)$.

$\mathsf{Eval}(crs, T, x, \pi) \to \pi'$: return $\hat{\Pi}.\mathsf{Eval}\left(crs, \hat{T}, x, \pi\right)$.

Under the above assumptions, Π is complete, zero-knowledge and CM simulation sound w.r.t. $\mathcal{T}$. Indeed, it suffices to define $\mathsf{SS\_Setup}$ as Setup only it returns sk as trapdoor, and Sim as an algorithm which signs x with sk and honestly proves knowledge of $(\perp, x, \mathrm{id}, \sigma)$. The witness indistinguishability of $\hat{\Pi}$ implies Π is zero-knowledge. The simulation soundness of Π stems from the fact that, since $\hat{\Pi}$ is sound, an adversary can win the game only if it can compute a valid signature on a word x_1' such that $(x_1', *)$ was never queried. However, as $\hat{\Pi}$ is partially extractable, that would contradict the existential unforgeability of SIG.

D.2 Strong Derivation Privacy.

In addition to the completeness, (CM simulation) soundness and zero-knowledge property for proof systems, a malleable proof system should also satisfy *derivation privacy*. It captures the idea that given $(x, w) \in \mathcal{R}$, proofs on a word $T_x(x)$ computed with Prove and $T_w(w)$ as a witness should be indistinguishable from those computed with Eval and a valid proof π for x. Note that re-randomizable malleable proofs necessarily satisfy derivation privacy [19, Theorem 2.7] (randomize then evaluate). An even *stronger* notion of derivation privacy is that the proof computed with Eval should be indistinguishable from those computed by the zero-knowledge simulator. Following a theorem of Chase et al. [19, Theorem 3.4], the above construction Π satisfies strong derivation privacy if $\hat{\Pi}$ satisfies derivation privacy.

D.3 Groth–Sahai Proofs

Groth and Sahai (GS) designed a practical, non-interactive witness-indistinguishable proof system for a wide class of statements in bilinear groups [28]. More precisely, their system allows to prove the existence of values which *simultaneously* satisfy pairing product equations, multi-scalar multiplication equations in the source groups, and quadratic equations in $\mathbb{Z}_p$. That is, given integers $m, n, m', n' \geq 1$, the GS proof system allows to prove that there exists $\mathbf{X} \in \mathbb{G}_1^m$, $\mathbf{Y} \in \mathbb{G}_2^n$, $\mathbf{x} \in \mathbb{Z}_p^{m'}$ and $\mathbf{y} \in \mathbb{Z}_p^{n'}$ which satisfy sets of equations of the form

$$\prod_{i=1}^{n} e\left(A_i, Y_i\right) \prod_{i=1}^{m} e\left(X_i, B_i\right) \prod_{i=1}^{m} \prod_{j=1}^{n} e\left(X_i, Y_i\right)^{\gamma_{ij}} = Z_T$$

$$\prod_{i=1}^{n'} \tilde{A}_i^{y_i} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n'} X_i^{\tilde{\gamma}_{ij} y_j} = Z_1$$

$$\prod_{i=1}^{n} Y_i^{a_i} \prod_{i=1}^{m'} \tilde{B}_i^{x_i} \prod_{i=1}^{m'} \prod_{j=1}^{n} Y_j^{\hat{\gamma}_{ij} x_i} = Z_2$$

$$\sum_{i=1}^{n'} \tilde{a}_i y_i + \sum_{i=1}^{m'} \tilde{b}_i x_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \gamma'_{ij} x_i y_j = z \mod p,$$

with the other values being public. The set of public values for which all sets of equations are satisfiable is further denoted $\mathcal{L}_{\mathrm{GS}}$. Multiscalar multiplication equations in G_2 are further omitted (i.e., $a_i := 0, \tilde{B}_i := 1_{\mathbb{G}_2}, \hat{\gamma}_{ij} := 0, Z_2 := 1_{\mathbb{G}_2}$) as there are not of interest in this paper, and as they are purely symmetric to equations in $\mathbb{G}_1$.

Their construction is given in the Common-Reference String (CRS) model. The CRS is generated in either of two modes: a *soundness mode* in which case the system is perfectly sound, and a *witness-indistinguishability mode* in which case the system is perfectly witness-indistinguishable. Under standard assumptions over bilinear groups, the two types of CRSs are indistinguishable.

Instantiation under the SXDH Assumption.

Common Reference String. In the instantiation under the SXDH assumption, the GS CRS contains two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1^2$ and two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{G}_2^2$. In a soundness setting, $\mathbf{b} \in \text{span}(\mathbf{a})$ and $\mathbf{w} \in \text{span}(\mathbf{v})$. Given the discrete-logarithm relation between a_2 and a_1, group-element variables can actually be efficiently extracted (which is not the case for $\mathbb{Z}_p$ elements as the discrete-logarithm problem is hard). In a witness-indistinguishable setting, $\mathbf{a}$ and $\mathbf{b}$ are linearly independent ($\mathbf{b} \leftarrow \mathbf{a}^t \begin{bmatrix} 1 & g_2^{-1} \end{bmatrix}^{\mathrm{T}}$ for $t \leftarrow_{\$} \mathbb{Z}_p$), and so are $\mathbf{v}$ and $\mathbf{w}$. In either setting, define $\mathbf{d} := \mathbf{b} \begin{bmatrix} 1 & g_2 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{u} := \mathbf{w} \begin{bmatrix} 1 & g_2 \end{bmatrix}^{\mathrm{T}}$.

Proof Computation. To compute a GS proof of satisfiability of equations of the form above,

- commit to all variables, i.e.,
  - for all $i \in [\![m]\!]$, generate $\rho_i, \sigma_i \leftarrow_{\$} \mathbb{Z}_p$
  - $\mathbf{c}(X_i) \leftarrow \begin{bmatrix} 1 & X_i \end{bmatrix}^{\mathrm{T}} \mathbf{a}^{\rho_i} \mathbf{b}^{\sigma_i}$
  - for all $i \in [\![n]\!]$, generate $\rho_i', \sigma_i' \leftarrow_{\$} \mathbb{Z}_p$
  - $\mathbf{c}(Y_i) \leftarrow \begin{bmatrix} 1 & Y_i \end{bmatrix}^{\mathrm{T}} \mathbf{v}^{\rho_i'} \mathbf{w}^{\sigma_i'}$
  - for all $i \in [\![m']\!]$, generate $r_i \leftarrow_{\$} \mathbb{Z}_p$
  - $\mathbf{c}(x_i) \leftarrow \mathbf{d}^{x_i} \mathbf{a}^{r_i}$
  - for all $i \in [\![n']\!]$, generate $r_i' \leftarrow_{\$} \mathbb{Z}_p$
  - $\mathbf{c}(y_i) \leftarrow \mathbf{u}^{y_i} \mathbf{v}^{r_i'}$
- compute proof elements for each pairing product equation, i.e.,
  - generate $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_p^{2\times 2}$
  - $\Theta \leftarrow \begin{bmatrix} \Theta_1 & \Theta_2 \end{bmatrix} =: \begin{bmatrix} 1 & \prod_i A_i^{\rho_i'} \prod_{i,j} X_i^{\gamma_{i,j}\rho_j'} \\ 1 & \prod_i A_i^{\sigma_i'} \prod_{i,j} X_i^{\gamma_{i,j}\sigma_j'} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} a_1^{T_{1,1}} b_1^{T_{1,2}} & a_2^{T_{1,1}} b_2^{T_{1,2}} \\ a_1^{T_{2,1}} b_1^{T_{2,2}} & a_2^{T_{2,1}} b_2^{T_{2,2}} \end{bmatrix}^{\mathrm{T}}$
  - $\Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix} =: \begin{bmatrix} 1 & \prod_i B_i^{\rho_i} \prod_{i,j} Y_j^{\gamma_{i,j}\rho_i} \\ 1 & \prod_i B_i^{\sigma_i} \prod_{i,j} Y_j^{\gamma_{i,j}\sigma_i} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} \mathbf{v}^{\sum_{i,j} \rho_i \gamma_{ij} \rho_j' - T_{1,1}} \mathbf{w}^{\sum_{i,j} \rho_i \gamma_{ij} \sigma_j' - T_{2,1}} \\ \mathbf{v}^{\sum_{i,j} \sigma_i \tilde{\gamma}_{ij} \rho_j' - T_{1,2}} \mathbf{w}^{\sum_{i,j} \sigma_i \gamma_{ij} \sigma_j' - T_{2,2}} \end{bmatrix}$
- for each multi-scalar multiplication equation in $\mathbb{G}_1$,
  - generate $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_p^2$
  - $\Theta \leftarrow \begin{bmatrix} 1 & \prod_i A_i^{r_i'} \prod_{i,j} X_i^{\tilde{\gamma}_{i,j} r_j'} \end{bmatrix}^{\mathrm{T}} \circ \begin{bmatrix} a_1^{T_1} b_1^{T_2} & a_2^{T_1} b_2^{T_2} \end{bmatrix}^{\mathrm{T}}$
  - $\Pi \leftarrow \begin{bmatrix} \Pi_1 \\ \Pi_2 \end{bmatrix} =: \begin{bmatrix} \mathbf{u}^{\sum_i b_i \rho_i + \sum_{i,j} \rho_i \tilde{\gamma}_{ij} y_j} \\ \mathbf{u}^{\sum_i b_i \sigma_i + \sum_{i,j} \sigma_i \tilde{\gamma}_{ij} y_j} \end{bmatrix} \circ \begin{bmatrix} \mathbf{v}^{\sum_{i,j} \rho_i \tilde{\gamma}_{ij} r_j' - T_1} \\ \mathbf{v}^{\sum_{i,j} \sigma_i \tilde{\gamma}_{ij} r_j' - T_2} \end{bmatrix}$

In case all $\tilde{\gamma}_{ij}$ are nil, then $\mathbf{T}$ can be set as $\mathbf{0}$, and the proof element Θ can then be restricted to its second component. Besides, if all $b_i = 0$, then Π can be set as $\mathbf{1}$.

- for each quadratic equation in $\mathbb{Z}_p$,
  - generate $T \leftarrow_{\$} \mathbb{Z}_p$
  - $\Theta \leftarrow \mathbf{d}^{\sum_i \tilde{a}_i r'_i + \sum_{i,j} r'_i \gamma'_{ij} y_j} \mathbf{a}^T$
  - $\Pi \leftarrow \mathbf{u}^{\sum_i \tilde{b}_i \rho_i + \sum_{i,j} \rho_i \gamma'_{ij} y_j} \mathbf{v}^{\sum_{i,j} \rho_i \gamma'_{ij} r'_j - T}$

  In case all γ'_{ij} are nil, then $\mathbf{T}$ can be set as $\mathbf{0}$. Moreover, Θ can also rather be set as $\sum_i \tilde{a}_i r'_i \in \mathbb{Z}_p$ and Π as $\sum_i \tilde{b}_i \rho_i \in \mathbb{Z}_p$, which results in smaller proof elements (the verification algorithm would then first compute $\mathbf{d}^{\Theta}$ and $\mathbf{u}^{\Pi}$).
- return all commitments $(\mathbf{c}(X_i))_{i=1}^{m}, (\mathbf{c}(Y_i))_{i=1}^{n}, (\mathbf{c}(x_i))_{i=1}^{m'}, (\mathbf{c}(y_i))_{i=1}^{n'}$ and all proof elements. Note that the resulting proof is perfectly *re-randomizable*, and that the proof system therefore satisfies derivation privacy.

Verification. Accept a proof parsed as above if and only if

- for every pairing production equation,

$$\begin{aligned}&\prod_i f\left(\begin{bmatrix}1\\A_i\end{bmatrix}, \mathbf{c}(Y_i)\right) \prod_i f\left(\mathbf{c}(X_i), \begin{bmatrix}1\\B_i\end{bmatrix}\right) \prod_{i,j} f\left(\mathbf{c}(X_i), \mathbf{c}(Y_j)\right)^{\gamma_{ij}}\\ =&\begin{bmatrix}1 & 1\\ 1 & Z_T\end{bmatrix} f\left(\Theta_1, \mathbf{v}\right) f\left(\Theta_1, \mathbf{w}\right) f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right)\end{aligned}$$

- for every multi-scalar multiplication equation in $\mathbb{G}_1$,

$$\begin{aligned}&\prod_i f\left(\begin{bmatrix}1\\\tilde{A}_i\end{bmatrix}, \mathbf{c}(y_i)\right) \prod_i f\left(\mathbf{c}(X_i), \begin{bmatrix}1\\b_i\end{bmatrix}\right) \prod_{i,j} f\left(\mathbf{c}(X_i), \mathbf{c}(y_j)\right)^{\tilde{\gamma}_{ij}}\\ =&f\left(\begin{bmatrix}1\\Z_1\end{bmatrix}, \mathbf{u}\right) f\left(\Theta, \mathbf{v}\right) f\left(\mathbf{a}, \Pi_1\right) f\left(\mathbf{b}, \Pi_2\right).\end{aligned}$$

- for every quadratic equation in $\mathbb{Z}_p$,

$$\begin{aligned}&\prod_i f\left(\mathbf{d}^{\tilde{a}_i}, \mathbf{c}(y_i)\right) \prod_i f\left(\mathbf{c}(x_i), \mathbf{u}^{\tilde{b}_i}\right) \prod_{i,j} f\left(\mathbf{c}(x_i), \mathbf{c}(y_j)\right)^{\gamma'_{ij}}\\ =&f\left(\mathbf{d}, \mathbf{u}^{z}\right) f\left(\Theta, \mathbf{v}\right) f\left(\mathbf{a}, \Pi\right).\end{aligned}$$

Malleability. Additive transformations in $\mathbb{Z}_p$ are admissible for the relation of multi-scalar multiplication equations in $\mathbb{G}_1$. That is, for any set $J \subseteq [\![n']\!]$ such that $\gamma_{ij} = 0$ for all $i \in [\![m']\!]$ and $j \in J$, given $(\alpha_i)_{i \in J} \in \mathbb{Z}_p^{|J|}$ and ℓ multi-scalar multiplication equations, the map

$$T_{(\alpha_i)_i} \colon \left(\left(\ldots, Z_{k,1}\right)_{k=1}^{\ell}, \left((y_i)_i, \ldots\right)\right) \mapsto \left(\left(\ldots, Z_{k,1} \prod_i \tilde{A}_{k,i}^{\alpha_i}\right)_k, \left((y_i + \alpha_i)_i, \ldots\right)\right)$$

is admissible. This map is identified with the tuple $(\alpha_i)_{i\in J}$. Note that it is the identity at the coordinates i such that $\alpha_i = 0$.

The GS proof system is malleable w.r.t. to such transformation. More precisely, its algorithm Eval first parses a valid π as above, generates $r_i'' \leftarrow_\$ \mathbb{Z}_p$, and computes

- $\mathbf{c}(y_i + \alpha_i) \leftarrow \mathbf{c}(y_i)\mathbf{u}^{\alpha_i}\mathbf{v}^{r_i''}$
- for all $k \in [\![\ell]\!]$, $\Theta_{k,2} \leftarrow \Theta_{k,2} \cdot \prod_i \tilde{A}_{ki}^{r_i''}$.

It then returns $\left((\mathbf{c}(X_i))_{i=1}^m,\ \mathbf{c}(y_i+\alpha_i)_{i=1}^{n'},\ (\Theta_k, \Pi_k)_{k=1}^\ell\right)$.

OR Proofs. Define $\mathcal{L}_{\text{OR-GS}}$ as the language of pairs (S_0, S_1) such that $S_0 \in \mathcal{L}_{\text{GS}}$ or $S_1 \in \mathcal{L}_{\text{GS}}$. To prove a statement $(S_0, S_1) \in \mathcal{L}_{\text{OR-GS}}$, following techniques of Groth [27], it suffices to introduce new variables $\chi_0 \in \mathbb{Z}_p$, and $\mathbf{X}_\beta \in \mathbb{G}_1^m$, $\mathbf{Y}_\beta \in \mathbb{G}_2^n$, $\mathbf{x}_\beta \in \mathbb{Z}_p^{m'}$ and $\mathbf{y}_\beta \in \mathbb{Z}_p^{n'}$ for $\beta \in \{0,1\}$, and then GS prove the satisfiability of the sets of equations

$$\prod_{i=1}^{n_\beta} e\left(A_{\beta,i}, Y_{\beta,i}\right) \prod_{i=1}^{m_\beta} e\left(X_{\beta,i}, B_{\beta,i}\right) \prod_{i=1}^{m_\beta}\prod_{j=1}^{n_\beta} e\left(X_{\beta,i}, Y_{\beta,i}\right)^{\gamma_{\beta,ij}} = Z_{\beta,T}^{\beta\chi_0+(1-\beta)(1-\chi_0)}$$

$$\prod_{i=1}^{n'_\beta} \tilde{A}_{\beta,i}^{y_{\beta,i}} \prod_{i=1}^{m_\beta} X_{\beta,i}^{b_{\beta,i}} \prod_{i=1}^{m_\beta}\prod_{j=1}^{n'_\beta} X_{\beta,i}^{\tilde{\gamma}_{\beta,ij}y_j} = Z_{\beta,1}^{\beta\chi_0+(1-\beta)(1-\chi_0)}$$

$$\sum_{i=1}^{n'_\beta} \tilde{a}_{\beta,i}y_{\beta,i} + \sum_{i=1}^{m'_\beta} \tilde{b}_{\beta,i}x_{\beta,i} + \sum_{i=1}^{m'_\beta}\sum_{j=1}^{n'_\beta} \gamma'_{\beta,ij}x_{\beta,i}y_{\beta,j} = z_\beta^{\beta\chi_0+(1-\beta)(1-\chi_0)} \mod p$$

$$\chi_0^2 - \chi_0 = 0 \mod p$$

for $\beta \in \{0,1\}$, and with $A_{0,i}$ denoting the variable A_i for S_0 and $A_{1,i}$ the one for S_1, and likewise for the other public values. If $S_0 \in \mathcal{L}_{\text{GS}}$ (with witness $(\mathbf{X}, \ldots)$), set $\mathbf{X}_0 \leftarrow \mathbf{X}$ and $\mathbf{X}_1 \leftarrow \mathbf{1}$, $\mathbf{x}_0 \leftarrow \mathbf{x}$ and $\mathbf{x}_1 \leftarrow \mathbf{0}$, and similarly for the other variables; and vice versa if $S_1 \in \mathcal{L}_{\text{GS}}$. The last equation implies that $\chi_0 = 0$ or 1 mod p, which guarantees that $S_0 \in \mathcal{L}_{\text{GS}}$ or $S_1 \in \mathcal{L}_{\text{GS}}$. To prove the last equation, it suffice to consider χ_0 as both an extra x and an extra y variable such that $x - y = 0 \mod p$ and $x - xy = 0 \mod p$.

Simulation Soundness under Controlled Malleability. The generic construction of a CM simulation proof system of Sec. D.1 can be applied with Jutla and Roy's signature scheme (App. A.2) to the GS proof system for multi-scalar multiplication equations in $\mathbb{G}_1$; the latter being malleable w.r.t. additive transformations in $\mathbb{Z}_p$.

In more detail, given equations of the form

$$\prod_{i=1}^{n'} \tilde{A}_{k,i}^{y_i} \prod_{i=1}^{m} X_i^{b_{k,i}} \prod_{i=1}^{m}\prod_{j=1}^{n'} X_i^{\tilde{\gamma}_{k,ij}y_j} = Z_{k,1}$$

for $1 \leq k \leq \ell$, let $\mathcal{T}$ denote the class of additive transformations. These transformations are identified with tuples $(\alpha_i)_i$. For a given transformation $(\alpha_i)_i$, let $K \subseteq [\![\ell]\!]$ denote the set of indices such that $Z_{k,1} \neq Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}$. Set μ as $\left(Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}\right)_{k \in K}$. The components of μ are nothing but the public values which are affected by the transformation (i.e., it corresponds to x_1' in the generic construction). A CM simulation sound GS proof for multi-scalar multiplication equations in $\mathbb{G}_1$ is then a proof that there exists $\mathbf{X}$ and $\mathbf{y}$ such that all ℓ equations are satisfied *or* that there exists μ, $(\alpha_i)_i$ and a Jutla–Roy signature σ such that $\mathsf{Vf}(vk, \mu, \sigma) = 1$ and $T_{(\alpha_i),x}\left(\left(\ldots, Z_{k,1} \prod_i \tilde{A}_{k,i}^{-\alpha_i}\right)_{k=1}^{\ell}\right) = (\ldots, Z_{k,1})_{k=1}^{\ell}$.

Compared to a standard GS proof, a CM-simulation-sound GS proof introduces $|\mu|$ additional variables $\mu_i \in \mathbb{G}_1$, 5 additional $\mathbb{G}_1$ variables and 1 additional $\mathbb{G}_2$ variables for σ, $|\mu|$ additional variables $\alpha_i \in \mathbb{Z}_p$ for the transformation, and 2 additional $\mathbb{Z}_p$ variables x and y (which should satisfy $x - y = x - xy = 0$).

In terms of equations, it introduces 2 pairing-product equations for the verification of the signature, $|\mu|$ multi-scalar multiplication equations in $\mathbb{G}_1$ for the transformation, and 2 quadratic equations in $\mathbb{Z}_p$ for x and y. That means $2(5+|\mu|)$ $\mathbb{G}_1$ and $2(3+|\mu|)$ $\mathbb{G}_2$ elements to commit to the additional variables; 4 $\mathbb{G}_1$ and 8 $\mathbb{G}_2$ elements as proof elements for the verification of the signature (which costs $4\mathrm{P}^6 + 4\mathrm{P}^{7+|\mu|}$); 2 $\mathbb{G}_1$, 2 $\mathbb{G}_2$ and 2 $\mathbb{Z}_p$ elements as proof elements for the two equations in $\mathbb{Z}_p$, the of which verification costs 2 $\mathbb{G}_1^1$ + 6 $\mathbb{G}_2^1$ + $8\mathrm{P}^5$; and $|\mu|$ $\mathbb{G}_1$ and $4|\mu|$ $\mathbb{G}_2$ elements for the multi-scalar multiplication equations in $\mathbb{G}_1$ introduced by μ, the verification of each of which costs $2\mathrm{P}^{5+|\{i\colon \tilde{A}_{k,i} \neq 1_{\mathbb{G}_1}\}|}$.

E Key-Derivation Functions

This section gives the formal security definition of key-derivation functions, and recalls Krawczyk's scheme [35].

E.1 Security

The security of key-derivation functions can be defined w.r.t. to a specific source of keying material which returns a material *SKM* and a public piece of information *inf*.

Definition E.1 ([35, Definition 7]). *A key-derivation function* KDF*, which supports salt values from a finite set* Σ*, is secure w.r.t. a source (of key material)*

SKM *if for every efficient adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} b = b' : \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{l} (SKM, inf) \leftarrow \mathsf{SKM}; Q \leftarrow \emptyset \\ XTS \leftarrow_{\$} \Sigma \\ (st, CTX^*, \ell^*) \leftarrow \mathcal{A}^{O.\mathsf{KDF}(Q,SKM,XTS,\cdot)}(inf, XTS) \\ b \leftarrow_{\$} \{0,1\} \\ \textit{if } b = 0 \\ \quad K^* \leftarrow_{\$} \{0,1\}^{\ell} \\ \textit{else} \\ \quad K^* \leftarrow \mathsf{KDF}(SKM, XTS, CTX^*, \ell^*) \\ b' \leftarrow \mathcal{A}^{O.\mathsf{KDF}(Q,SKM,XTS,\cdot)}(st, K^*) \\ \textit{if } CTX^* \in Q \\ \quad b' \leftarrow_{\$} \{0,1\} \\ \quad \textit{return } (b, b') \\ \textit{return } (b, b') \end{array}\right]$$

is negligibly close to 1/2, with $O.\mathsf{KDF}$ *an oracle which, on input* (SKM, XTS)*, replies to a* (CTX, ℓ) *query with* $\mathsf{KDF}(SKM, XTS, CTX, \ell)$ *and then adds* CTX *to* Q*.*

The previous definition is w.r.t. to a specific source, but it can be extended to all sources which return materials with enough entropy even when conditioned on the public information. Formally, for an integer m, a source SKM is a m-entropy source if for all s in the range of key materials and all i in the support of public information returned by SKM, $\Pr[SKM = s | inf = i] \leq 2^{-m}$. A key-derivation function is then said to be *m-entropy secure* if it is secure w.r.t. all m-entropy sources.

E.2 Krawczyk's Key-Derivation Function

Krawczyk gave [35, Section 4.2] a secure construction of KDFs from Hash-based Message Authentication Codes [6] (HMAC) which follows the extract-then-expand paradigm.

More precisely, let $\{H_\kappa\}_{\kappa \in \{0,1\}^k}$ be a family of Merkle–Damgård hash functions with k-bit outputs which is based on a family of compression functions $\{h_\kappa\}_\kappa$ (think of SHA-512). Assume (single-keyed) HMAC to be built from NMAC [6], and NMAC to be built from $\{H_\kappa\}_\kappa$. If $\{h_\kappa\}_\kappa$ is a family of pairwise-independent compression functions, and $\{H_\kappa\}_\kappa$ is collision-resistant against linear-size circuits, then [35, Corollary 9] NMAC truncated by c bits is a $\left(m, (n+2)2^{-c/2}\right)$-statistical extractor on n-block inputs. If $\{h_\kappa\}_\kappa$ is modeled as a family of random functions independent from the source, then the same result applies to HMAC.

Moreover, given another family of compression functions $\{g_\kappa\}_\kappa$ (think of SHA-256), if $g_\cdot$ is a Pseudo-Random Function (PRF), and if $\hat{g}\colon (K, M) \to g_M(K)$ is a PRF under a class of affine related-key attacks (defined by the inner and outer pads), then [5, Theorem 3.3, Lemma 5.2] HMAC is a (q_P, ε_P)-PRF for q_P, ε_P explicited in Bellare's paper [5].

Krawczyk's theorem [35, Theorem 1] implies that HMAC is a $(q_P, \varepsilon_P + (n+2)\, 2^{-c/2})$-secure KDF w.r.t. to sources with min-entropy at least m. Throughout the paper, it is assumed that n and c are functions of λ such that $n2^{-c/2}$ is negligible in λ. The construction is described in Algorithm 1. Therein, given two integers $n \geq d \geq 1$ and $x \in \{0,1\}^n$, $[x]_d$ denotes the sub-string of x consisting of its first d bits.

Algorithm 1 Krawczyk's HMAC-based KDF.

Require: (SKM, XTS, CTX, L) and HMAC based on a hash function with output length k
Ensure: $K \in \{0,1\}^L$
  $t \leftarrow \lceil L/k \rceil$
  $d \leftarrow L \bmod k$
  $PRK \leftarrow \mathsf{HMAC}(XTS, SKM)$
  $K(1) \leftarrow \mathsf{HMAC}(PRK, CTX \| 0)$
  for $i = 1$ to $t-1$ **do**
    $K(i+1) \leftarrow \mathsf{HMAC}(PRK, K(i) \| CTX \| i)$
  end for
  $K \leftarrow K(1) \| \cdots \| [K(t)]_d$
  return K

F Security of the Construction

This section proves the security of the construction in Section 4.

F.1 Security Proofs

Recall from Sec. 4.2 that the P-IND-RCCA security and the verifiabilty of $\mathcal{E}$ rely of the following assumptions:

- $\{h_\kappa\}_\kappa$ is pairwise-independent
- $\{H_\kappa\}_\kappa$ is collision-resistant against linear-size circuits
- $g.$ is a secure Pseudo-Random Function (PRF)
- $\hat{g}\colon (K, M) \to g_M(K)$ is a secure PRF under a class of affine related-key attacks defined by the inner and outer pads [5, Lemma 5.2].

Theorem F.1 (P-IND-RCCA). *Assuming that the SXDH assumption over* G *holds, that* $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, $g.$ *and* $\hat{g}.$ *satisfy the assumptions above, that* $\mathcal{H}_{\mathrm{PCA}}$ *is second-preimage resistant and that* $\mathcal{H}_{\mathrm{OTS}}$ *is collision-resistant,* $\mathcal{E}$ *is P-IND-RCCA secure.*

Proof. Let $\mathcal{A}$ be an adversary for the P-IND-RCCA game which makes at most q_{Exec} Exec queries and at most q_{Send} Send queries. A message is subsequently

said to be *oracle-generated* if it was computed by the challenger as a response to a Send query and it was not altered. Otherwise, the message is said to be *adversarially generated*. If an adversarially generated message is given as an input to an algorithm which simply forwards it (possibly after some verifications), the message is still considered adversarially generated. Recall that only static corruptions are considered. However, $\mathcal{A}$ can of course modify the messages sent between the parties. Recall also that all instances are assumed to erase their temporary variables (which include their randomness) upon termination.

Further distinguish the following cases:

1. $\mathcal{A}$ never makes a Send query to an $\mathcal{S}$ instance on a valid tuple (i.e., which passes all verification) $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated but $(C_{\mathcal{T}}, ovk)$ is adversarially generated, although $\mathcal{T}$ and $\mathcal{S}$ are honest.
2. $\mathcal{A}$ makes a Send query of the above form.

In the first case, the main idea is to define a game which is indistinguishable from the real one, but in which the adversary cannot modify the messages computed by honest parties within a session; unless it knows their passwords in the case of user and server identities. Moreover, the oracle-generated messages in this latter game are independent of the passwords, so as long as a user and one of her servers are honest, $\mathcal{A}$ can only guess their common password, and it can only do so with probability at most $H_{\infty,\mathsf{PG}}$. Winning the 1-out-of-2 IND-RCCA security of $\mathcal{E}_{\mathsf{rcca}}$ can then be reduced to winning that game.

To this end, consider the sequence of games hereafter. It starts by modifying how Exec queries are handled, and then continues with Send queries.

Game 0. This is the real P-IND-RCCA game.

Game 1. In this game, the challenger generates trapdoor SS_GS parameters, i.e., a signing key sk which allows to simulate proofs, as well as the discrete-logarithm relations between $\tilde{a}_1$ and $\tilde{a}_2$, and between $\tilde{b}_1$ and $\tilde{b}_2$, which allow to extract committed group elements. As the resulting parameters are perfectly indistinguishable from honest parameters, Game 1 and Game 0 are perfectly indistinguishable.

Game 2. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger simulates $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$. Note that since SS_GS satisfies perfect strong derivation privacy, proofs in the previous game (leveraging its malleability) are perfectly indistinguishable from proofs computed with the full secret key. By the zero-knowledge property of SS_GS, these latter are indistinguishable from simulates proofs of the full secret key.

The zero-knowledge property of SS_GS implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Exec}})$, with the latter denoting the supremal advantage of any PPT adversary in distinguishing real proofs from simulated ones.

Note that even if $\mathcal{S}$ is corrupt at the beginning of the protocol and $\mathcal{A}$ thus knows the witness of the proofs, it still cannot distinguish real proofs

from simulated ones since $\mathsf{SS\_GS}$ is zero-knowledge. Moreover, as temporary variables of the token algorithm instance are erased upon termination, $\mathcal{A}$ cannot distinguish real proofs from simulated ones by corrupting $\mathcal{S}$ after the execution either.

Game 3. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger computes $H_{\mathcal{U}} = H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}\right) \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}\right)$. Game 3 from Game 2 are perfectly indistinguishable by correctness of the SPHF.

Game 4. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger now computes $C_{\mathcal{U}}$ as $\mathsf{Enc}^{hp_{\mathcal{U}}}\left(ek, 1_{\mathbb{G}_1}\right)$ and $C_{\mathcal{S}}$ as $\mathsf{Enc}^{hp_{\mathcal{S}}}\left(ek, 1_{\mathbb{G}_1}\right)$ (recall that $1_{\mathbb{G}_1}$ is assumed not to be a valid password).

The indistinguishability of Game 4 from Game 3 stems from the IND-CPA security (implied by the IND-PCA security) of $\mathcal{E}_{\mathsf{pca}}$ which relies on the SXDH assumption. The distinguishing advantage of $\mathcal{A}$ is at most $q_{\mathsf{Exec}}\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$, with $\varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$ denoting the supremal advantage of any efficient adversary in the IND-CPA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 5. The challenger now answers Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$ by choosing $H_{\mathcal{U}} = H_{\mathcal{S}}$ uniformly at random. This game is perfectly indistinguishable from the previous one by the smoothness property of the SPHF.

Game 6. In this game, the challenger also saves the short-Cramer–Shoup decryption key dk. Game 6 is perfectly indistinguishable from Game 5.

Game 7. To answer Send queries to an $\mathcal{S}$ instance on $\left(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}}\right)$ and on $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$, the challenger simulates $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$. The challenger also simulates $\tilde{\pi}_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to answer queries to $\mathcal{T}$ instances. The same arguments as in Game 2 imply that $\mathcal{A}$ can distinguish Game 7 from Game 6 with an advantage of at most $\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}})$.

Game 8. The challenger now answers Send queries on $\left(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}}\right)$ to an $\mathcal{S}$ instance as follows:

- if $\mathsf{OTS.Vf}\left(ovk, \left(C_{\mathcal{U}}, hp_{\mathcal{U}}, \tilde{\pi}_{\mathcal{T}}\right), \sigma_{\mathcal{T}}\right) \neq 1$ or $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) \neq pk_2 \, pk_1^{-\alpha_{1,\mathcal{S}}} g_1^{-\alpha_{2,\mathcal{S}}}$ or $\mathsf{GS.Vf}\left(crs, pk_{\mathcal{U}}, \alpha_{i,\mathcal{S}}, \tilde{\pi}_{\mathcal{T}}\right) \neq 1$, return $\perp$
- if $\left(C_{\mathcal{U}}, hp_{\mathcal{U}}\right)$ is oracle-generated and the tuple of the Send query also is, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}\right) \cdot \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}\right)$ (the challenger knows $hk_{\mathcal{U}}$ since $\left(C_{\mathcal{U}}, hp_{\mathcal{U}}\right)$ is honestly generated). In this case, this game is perfectly indistinguishable from the previous one by the correctness of the SPHF
- if $\left(C_{\mathcal{U}}, hp_{\mathcal{U}}\right)$ and $\left(C_{\mathcal{T}}, ovk\right)$ are oracle-generated (i.e., $(*, ovk, *, C_{\mathcal{T}}, *)$ was computed as an answer to a Send query) but the tuple is adversarially generated, if $\mathcal{T}$ is honest, abort as $\mathcal{A}$ contradicted the strong one-time security of OTS. If $\mathcal{T}$ is corrupt, then reply with $\perp$ if $\mathcal{U}$ is corrupt; if $\mathcal{U}$ is honest, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}\right) \cdot \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}\right)$.

  Denoting by $\varepsilon_{\mathsf{OTS}}$ the supremal advantage of any efficient adversary in the strong one-time security game, $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $|I|\varepsilon_{\mathsf{OTS}}$ (the reduction al-

gorithm must guess the $\mathcal{T}$ instance for which it sets the one-time signing key as that of the challenger)

- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated but $(C_{\mathcal{T}}, ovk)$ is adversarially generated,
  - if $\mathcal{T}$ is corrupt then
    - if $(\mathcal{U}, \mathcal{T}, \mathcal{S}) = (\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ and $\mathcal{U}^*$ is corrupt, reply with $\perp$ thereby following the definition of oracle Send
    - else, compute $H_{\mathcal{S}} \leftarrow \mathsf{Hash}(hk_{\mathcal{U}},\ \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}) \cdot \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}},\ C_{\mathcal{U}})$
  - if $\mathcal{T}$ is honest, then $\mathcal{S}$ is necessarily corrupt as $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated and $(C_{\mathcal{T}}, ovk)$ is adversarially generated (cf. conditions of case 1). Compute $H_{\mathcal{S}}$ as in the real game
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is adversarially generated and $\mathcal{U}$ and $\mathcal{S}$ are honest, check whether $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{U}}}(dk, C_{\mathcal{U}}) = p_{\mathcal{S}}^{\mathcal{U}}$. If so, i.e., $\mathcal{A}$ correctly guessed $p_{\mathcal{S}}^{\mathcal{U}}$, abort and return 1 indicating that $\mathcal{A}$ won the game (which increases the advantage of $\mathcal{A}$ in this game). If not, generate $H_{\mathcal{S}}$ uniformly at random, and the perfect smoothness of the SPHF implies the perfect indistinguishability from the previous game
- if $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is adversarially generated and $\mathcal{U}$ or $\mathcal{S}$ is corrupt,
  - if $(\mathcal{U}, \mathcal{T}, \mathcal{S}) = (\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ and $\mathcal{T}^*$ is corrupt, return $\perp$. By definition of oracle Send, Game 8 is perfectly indistinguishable from Game 7
  - if $\mathcal{T}$ is honest, compute $H_{\mathcal{S}}$ as in the real game.

The advantage of $\mathcal{A}$ in the previous game is therefore upper-bounded by its advantage in Game 8 plus $q_{\mathsf{Send}}|I|\varepsilon_{\mathsf{OTS}}$.

Game 9. The challenger now answers Send on $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to a $\mathcal{U}$ instance as follows:

- if $\mathsf{SS\_GS.Vf}(crs, pk_{\mathcal{U}}, \tilde{\pi}_{\mathcal{S}}) \neq 1$, return $\perp$
- if $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is oracle-generated, compute $H_{\mathcal{U}} \leftarrow \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}) \cdot \mathsf{Hash}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}})$ (the challenger knows $hk_{\mathcal{S}}$ as $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is honestly generated). In this case, Game 9 is perfectly indistinguishable from Game 8 by the correctness of the SPHF
- if $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is adversarially generated and $\mathcal{S}$ and $\mathcal{U}$ are honest, check whether $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{S}}}(dk, C_{\mathcal{S}}) = p_{\mathcal{U}}$. If so (i.e., $\mathcal{A}$ correctly guessed $p_{\mathcal{U}}$), abort and return 1. If not, choose $H_{\mathcal{U}}$ uniformly at random, and the perfect smoothness of the SPHF implies the perfect indistinguishability from the previous game
- if the tuple is adversarially generated and $\mathcal{S}$ or $\mathcal{U}$ is corrupt, compute $H_{\mathcal{U}}$ as in the real game.

The advantage of $\mathcal{A}$ in the previous game is thus upper-bounded by its advantage in this game.

From this game onwards, the randomness used to encrypt passwords is necessary to compute neither $H_{\mathcal{U}}$ nor $H_{\mathcal{S}}$ in case $\mathcal{U}$ and $\mathcal{S}$ are honest.

Game 10. In this game, the challenger answers Send queries to

- an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}},\ C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated, and which are oracle-generated if $\mathcal{T}$ is honest, by generating $H_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$ in case $\mathcal{U}$ is honest

– a $\mathcal{U}$ instance on oracle-generated tuples $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$ by setting $H_{\mathcal{U}} \leftarrow H_{\mathcal{S}}$, the latter being the value of the partner $\mathcal{S}$ instance of the $\mathcal{U}$ in the current session

A lemma by Katz and Vaikuntanathan [32, Lemma 1] states that SPH values are computationally indistinguishable from uniformly random group elements even if hashing keys and ciphertexts are used several times and if projective keys made public, under the assumption that the encryption scheme is IND-PCA secure (they actually prove it in the case of IND-CCA security, but the application of the lemma only makes use of plaintext-check queries) and that the SPHF is statistically smooth. The lemma then entails that $\mathcal{A}$ can distinguish Game 10 from Game 9 with an advantage of at most $2q_{\mathsf{Send}}^2 \varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\text{ind-pca}}(q_{\mathsf{Send}})$, with $\varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\text{ind-pca}}(q_{\mathsf{Send}})$ denoting the supremal advantage of any efficient adversary which makes at most q_{Send} plaintext-check queries in the IND-PCA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 11. For prompting queries $(\mathcal{T}, *, \mathcal{S}, *, *)$ to a $\mathcal{U}$ instance and Send queries to an $\mathcal{S}$ instance on oracle-generated tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is also oracle-generated, if $\mathcal{U}$ and $\mathcal{S}$ are honest, the challenger computes $C_{\mathcal{S}}$ as $\mathsf{Enc}^{hp_{\mathcal{U}}}(ek, 1_{\mathbb{G}_1})$ and $C_{\mathcal{U}}$ as $\mathsf{Enc}^{hp_{\mathcal{S}}}(ek, 1_{\mathbb{G}_1})$. Distinguishing Game 11 from Game 10 can then be reduced to winning the IND-PCA game for $\mathcal{E}_{\mathsf{pca}}$.

Note that for any Send query on $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ to an $\mathcal{S}$ instance, if the tuple is oracle-generated and $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ also is, the reduction algorithm already knows the password encrypted in $C_{\mathcal{U}}$ and can then do the same test as the challenger of Game 8. Similarly for Send queries on tuples $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to $\mathcal{U}$ instances. In case $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ or $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is adversarially generated, the reduction can make a plaintext-check queries.

It follows that $\mathcal{A}$ can distinguish Game 11 from Game 10 with an advantage at most $2q_{\mathsf{Send}}\varepsilon_{\mathcal{E}_{\mathsf{pca}}}^{\text{ind-pca}}(q_{\mathsf{Send}})$. Note also that from this game on, the messages computed by instances of $\mathcal{U}$ and $\mathcal{S}$ are independent of the passwords if they are both honest, $\mathcal{A}$ can then guess their common password with probability at most $H_{\infty,\mathsf{PG}}$ at each Send query.

Game 12. For Send queries to an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated, and which are oracle-generated if $\mathcal{T}$ is honest, the challenger generates $K_{\mathcal{S}}$ uniformly at random in case $\mathcal{U}$ is honest. For a Send query to a partner $\mathcal{U}$ instance on an oracle-generated tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$, the challenger sets $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$.

Distinguishing this game from the previous can be reduced to the security of KDF w.r.t. uniformly random sources. Note that since the distribution of the source is independent of the adversary, generating the salt value XTS before the end of the PAKE does not raise any issue in the reduction. Indeed, the reduction algorithm can generate a uniformly random source value at the beginning of the reduction, submit it to the KDF-security-game challenger, then receive back a uniform salt value before generating the other parameters for $\mathcal{E}$.

Denoting by $\varepsilon_{\mathsf{KDF}}(0)$ the supremal advantage of any efficient adversary which makes no oracle query in the KDF security game with scheme KDF (cf. Appendix E), adversary $\mathcal{A}$ can distinguish Game 12 from Game 11 with an advantage at most $|I|^2\varepsilon_{\mathsf{KDF}}(0)$ (the reduction algorithm guesses the instances of $\mathcal{U}$ and $\mathcal{S}$ for which it sets the keys as the key returned by the KDF challenger).

Game 13. For a Send query on an adversarially generated tuple $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ to a $\mathcal{U}$ instance, abort the protocol if $\mathcal{S}$ and $\mathcal{U}$ are honest. Moreover, if the tuple is oracle-generated in a different session, abort the protocol. Likewise, for a Send query on a an adversarially generated tuple $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$ to an $\mathcal{S}$ instance, if $\mathcal{U}$ and $\mathcal{S}$ are honest, abort the protocol. Besides, if the tuple is oracle-generated in a different session, abort the protocol.

Distinguishing Game 13 from Game 12 can be reduced to the security of MAC. It follows $\mathcal{A}$ can distinguish Game 13 from Game 12 with an advantage of at most $|I|^2\varepsilon^{\mathrm{prf}}_{\mathsf{MAC}}(1)$, with $\varepsilon^{\mathrm{prf}}_{\mathsf{MAC}}(1)$ being the supremal advantage of any efficient adversary which makes at most one oracle query in the PRF game with scheme MAC (cf. Appendix E) (the reduction algorithm guesses the $\mathcal{U}$ and $\mathcal{S}$ instance for which it sets the common MAC key as the challenge one).

Game 14. For a Send query to a $\mathcal{T}$ instance on pair $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ which is either adversarially generated or oracle-generated in a different session, abort if $\mathcal{U}$ and $\mathcal{S}$ are honest. Indeed, the word for which $\pi_{\mathcal{S}}$ is a proof is generated anew for each session since $hk_{\mathcal{S}}$ is always freshly generated (and $hp_{\mathcal{S}}$ is thus not the image of a previous projective key under an additive transformation). It follows that

- if $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ is oracle generated in a different session, the verification can only succeed with negligible probability if $\mathsf{SS\_GS}$ is sound. In this case, $\mathcal{A}$ can distinguish this game from the previous with an advantage of at most $\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}})$, with the latter denoting the supremal advantage of any efficient adversary which makes at most q_{Send} queries in the CM-simulation-soundness game for $\mathsf{SS\_GS}$
- if $(M_{\mathcal{S}}, \pi_{\mathcal{S}})$ is adversarially generated, the CM simulation soundness of $\mathsf{SS\_GS}$ guarantees that the commitments $\mathbf{c}(\lambda)$, $\mathbf{c}(\mu)$, $\mathbf{c}(\nu)$, $\mathbf{c}(\theta)$ satisfy $f\left(g_1, \mathbf{c}(\lambda)\right) f\left(h_1, \mathbf{c}(\nu)\right) f\left(\gamma, \mathbf{c}(\theta)\right) = f\left(hp_1,\ \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},1}}, \mathbf{v}\right)$ and $f\left(g_1, \mathbf{c}(\mu)\right) f\left(\delta, \mathbf{c}(\theta)\right) = f\left(hp_2,\ \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},2}}, \mathbf{v}\right)$ (with proof elements excerpted from $\pi_{\mathcal{S}}$). However, these commitments can then be used to contradict the perfect smoothness of the KV-SPHF for short Cramer–Shoup ciphertext. Indeed, given $M \in \mathbb{G}_1$ and $C \coloneqq (U, E, V)$ such that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{hp_{\mathcal{S}}}(dk, C) \neq M$, these commitments and $hp_{\mathcal{S}}$ can be used to distinguish $\mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_M, C)$ from $\mathfrak{H} \leftarrow_{\$} \mathbb{G}_1$ simply by testing whether $f\left(U, \mathbf{c}(\lambda)\right) f\left(U^{\xi}, \mathbf{c}(\mu)\right) f\left(E/M, \mathbf{c}(\nu)\right) f\left(V, (\theta)\right) = f\left(\mathfrak{H}, \mathbf{u}\right) f\left(\Theta_{hp_{\mathcal{S},1}}\Theta^{\xi}_{hp_{\mathcal{S},2}}, \mathbf{v}\right)$. The perfect smoothness of the KV-SPHF for short Cramer–Shoup ciphertexts implies that it can only occur with probability $1/p$. Therefore, in this case, the adversary can distinguish this game from the previous one with an

advantage of at most $q_{\mathsf{Send}}\left(\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + p^{-1}\right)$ (the algorithm for the reduction to the perfect smoothness of the SPHF must guess the query for which it sets the projective key as $hp_{\mathcal{S}}$)

$\mathcal{A}$ can then distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Send}}\left(\varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + p^{-1}\right)$.

In Game 14, once the challenge tuple $(\mathcal{U}^*, \mathcal{T}^*, \mathcal{S}^*)$ is defined, all values received by $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$ instances up to $(M_{\mathcal{S}^*}, \pi_{\mathcal{S}^*})$ included are either all oracle-generated in the same session or replied to with $\perp$ as long as these identities are honest. In particular, if $\mathcal{U}^*$ is corrupt, Send queries to $\mathcal{T}^*$ instances are rejected (by definition of oracle Send) and $\mathcal{A}$ thus cannot make successful Send queries to $\mathcal{S}^*$ instances anymore as long as $\mathcal{T}^*$ is honest; and if $\mathcal{U}^*$ and $\mathcal{T}^*$ are corrupt, Send queries to $\mathcal{S}^*$ instances are rejected anyway. Likewise, if $\mathcal{S}^*$ is corrupt, Send queries to $\mathcal{T}^*$ instances are rejected (recall that $\mathcal{S}^*$ and $\mathcal{T}^*$ cannot both be corrupt) and $\mathcal{A}$ cannot make successful Send queries to $\mathcal{S}^*$ instances anymore. Winning the 1-out-of-2 RCCA game for $\mathcal{E}_{\mathsf{rcca}}$ can then reduced to winning Game 14 as follows.

At the beginning of the game, the reduction algorithm, further denoted $\mathcal{B}$, guesses the identities $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$. If $\mathcal{A}$ later makes its Test query on a different tuple of identities, $\mathcal{B}$ simply aborts and sends to the challenger a bit chosen uniformly at random.

Recall that $\mathcal{T}^*$ and $\mathcal{S}^*$ cannot both be corrupt by definition of P-IND-RCCA game. On this account, first suppose that $\mathcal{S}^*$ is honest throughout the game. $\mathcal{B}$ then sets the public key as the public key of $\mathcal{U}^*$ and asks for a secret key share that it sets as the share of $\mathcal{T}^*$ (which is given to $\mathcal{A}$ in case of corruption). The other share is then implicitly the share of party $\mathcal{S}^*$.

For Exec queries and Send queries on oracle-generated tuples $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}^*}\right)$ to $\mathcal{S}^*$ instances, since $\tilde{C}_M$ is computed by $\mathcal{B}$, the latter can make decryption queries to the RCCA challenger on the original ciphertext C_M, receive back $x_3 x_1^{-\alpha_{1,\mathcal{S}^*}} x_2^{-\alpha_{1,\mathcal{S}^*}}$, and multiply it by $RH_{\mathcal{S}^*}$, with R denoting the blinding factor generated by algorithm Blind.

$\mathcal{B}$ also simulates with the trapdoor proofs $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$.

For query Test, algorithm $\mathcal{B}$ simply forwards (M_0, M_1) to the RCCA challenger. After the test query, for Exec queries as above, if C_M decrypts to M_0 or M_1, the RCCA challenger answers with replay and so does $\mathcal{B}$.

For a prompting Send query on $(\mathcal{T}^*, *, \mathcal{S}^*, *, C_M)$ to a $\mathcal{U}^*$ instance, the reduction makes a decryption query to the RCCA challenger on C_M. If it decrypts to M_0 or M_1, the reduction algorithm receives replay and forwards it to $\mathcal{A}$.

The condition "if $(\mathsf{Dec}(sk_{\mathcal{U}}, C) \in \{M_0, M_1\})$ return replay" guarantees that $\mathcal{B}$ can answer with $\perp$ Exec queries and prompting Send queries on C^*, and perfectly emulate the Game-14 challenger.

Moreover, recall that in case 1), $\mathcal{A}$ never makes a Send query on a valid tuple $\left(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*}, ovk, \tilde{\pi}_{\mathcal{T}^*}, C_{\mathcal{T}^*}, \sigma_{\mathcal{T}^*}\right)$ such that $\left(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*}\right)$ is oracle-generated but $(C_{\mathcal{T}^*}, ovk)$ is adversarially generated, although $\mathcal{T}^*$ and $\mathcal{S}^*$ are honest. It means that if $\mathcal{U}^*$ is honest, $\mathcal{A}$ can obtain partial decryption from $\mathcal{S}^*$ only with

oracle-generated tuples, and the condition mentioned above ensures that $\mathcal{S}^*$ never has to make a decryption query on a ciphertext which decrypts to M_0 or M_1. If $\mathcal{U}^*$ is corrupt and $\mathcal{T}^*$ is honest, $\mathcal{B}$ never has to make such a decryption query either as $\mathcal{A}$ would never submit a valid tuple in the first flow. If $\mathcal{U}^*$ and $\mathcal{T}^*$ are both corrupt, then $\mathcal{B}$ can simply answer Send queries with $\perp$.

As $\mathcal{B}$ perfectly emulates the Game-14 challenger, it wins the RCCA game with at least the same advantage as $\mathcal{A}$ in that game.

In case $\mathcal{T}^*$ is honest throughout the game, the reduction is similar except that $\mathcal{B}$ now sets the key share it gets as the share of $\mathcal{S}^*$. Note that in Game 14, all valid pairs $(M_{\mathcal{S}^*}, \pi_{\mathcal{S}^*})$ submitted to $\mathcal{T}^*$ instances are oracle-generated in the same session if $\mathcal{U}^*$ and $\mathcal{S}^*$ are honest. If either of them is corrupt, all Send queries to $\mathcal{T}^*$ instances are rejected. Consequently, $\mathcal{B}$ never has to make a decryption query on a ciphertext that decrypts to M_0 or M_1.

It follows that the advantage of $\mathcal{A}$ in the P-IND-RCCA game in case 1) is at most

$$\begin{aligned}
&\varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Exec}}) + \varepsilon^{zk}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) + q_{\mathsf{Exec}}\varepsilon^{\text{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}} + |I|\varepsilon_{\mathsf{OTS}} \\
&+ q_{\mathsf{Send}}\left(H_{\infty,\mathsf{PG}} + \varepsilon^{\text{sound}}_{\mathsf{SS_GS}}(q_{\mathsf{Send}}) + p^{-1} + 2(q_{\mathsf{Send}}+1)\varepsilon^{\text{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})\right) \\
&+ |I|^2\left(\varepsilon_{\mathsf{KDF}}(0) + \varepsilon^{\text{prf}}_{\mathsf{MAC}}(1)\right) + |U||T||S|\mathsf{Adv}^{\text{ind-rcca}}_{\mathsf{G},\mathcal{E}_{\mathsf{rcca}}}(\lambda).
\end{aligned}$$

In the second case, $\mathcal{T}$ and $\mathcal{S}$ are honest in the query that distinguishes the two cases. The major argument is that only $\mathcal{T}$ and $\mathcal{S}$ can compute $pk_1^{\alpha_{1,\mathcal{T}}} g_1^{\alpha_{2,\mathcal{T}}}$ and that only $\mathcal{S}$ holds the decryption dk'. Therefore, an adversary cannot distinguish $C_{\mathcal{T}}$ from encryption of a dummy message which contains no information about the token share. Moreover, as long as $\mathcal{U}$ and $\mathcal{S}$ are honest, $H_{\mathcal{S}}$ is indistinguishable from a uniformly random value and $M_{\mathcal{S}}$ thus contains no information about the shares either. For the adversary to compute a valid tuple from that point, it has to compute a valid proof on the token share after only getting zero-knowledge ones. This valid proof can then be used to contradict the 1-out-of-2 security of $\mathcal{E}_{\mathsf{rcca}}$.

To prove it formally, consider the following sequence of games.

Game 0–5. These are the same as in the previous case.

Game 6. To answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger computes, the challenger generates $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$ uniformly at random. $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $|I|\varepsilon_{\mathsf{KDF}}(0)$.

Game 7. In this game, to answer Exec queries with identities $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{S}$, the challenger generates $M_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$. This game is perfectly indistinguishable from the previous one as $H_{\mathcal{S}}$ is uniformly random in the latter.

Game 8. To answer Exec queries with identities $\mathcal{U}, \mathcal{T}$ and $\mathcal{S}$, the challenger now computes $C_{\mathcal{T}}$ as $\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}\left(ek', 1_{\mathbb{G}_1}\right)$ (and verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = 1_{\mathbb{G}_1}$). The indistinguishability from the previous games stems from

the IND-CPA security of $\mathcal{E}_{\mathsf{pca}}$ (which is implies by its IND-PCA security). Therefore, $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Exec}}\varepsilon^{\text{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$, with $\varepsilon^{\text{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}}$ denoting the supremal advantage of any efficient adversary in the IND-CPA game with scheme $\mathcal{E}_{\mathsf{pca}}$.

Game 9. To compute $\tilde{\pi}_{\mathcal{S}}$ and $\pi_{\mathcal{S}}$ to answer Send queries to $\mathcal{S}$ instances, the challenger now simulates them with the trapdoor. Likewise, the challenger simulates $\tilde{\pi}_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$ to answers Send queries to $\mathcal{T}$ instances. The zero-knowledge property of $\mathsf{SS\_GS}$ implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}})$.

Game 10. The challenger now answers Send queries to an $\mathcal{S}$ instance on $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated by computing $H_{\mathcal{S}} \leftarrow \mathsf{Hash}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}) \cdot \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}})$. This game is perfectly indistinguishable from the previous one by the correctness of the SPHF.

Game 11. The challenger now answers Send to a $\mathcal{U}$ instance on $(C_{\mathcal{S}}, hp_{\mathcal{S}}, \tilde{\pi}_{\mathcal{S}}, \tau_{\mathcal{S}})$ such that $(C_{\mathcal{S}}, hp_{\mathcal{S}})$ is oracle-generated by computing $H_{\mathcal{U}} \leftarrow \mathsf{Hash}(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}) \cdot \mathsf{Hash}(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}})$. This game is perfectly indistinguishable from the previous one by the correctness of the SPHF.

Game 12. In this game, the challenger answers Send queries to

- an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated by generating $H_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$ in case $\mathcal{U}$ is honest
- a $\mathcal{U}$ instance on oracle-generated tuples $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$ by setting $H_{\mathcal{U}} \leftarrow H_{\mathcal{S}}$, the latter being the value of the partner $\mathcal{S}$ instance of the $\mathcal{U}$ in the current session

Katz and Vaikuntanathan's lemma [32, Lemma 1] implies that $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $2q^2_{\mathsf{Send}}\varepsilon^{\text{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})$.

Game 13. For Send queries to an $\mathcal{S}$ instance on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$ such that $(C_{\mathcal{U}}, hp_{\mathcal{U}})$ is oracle-generated, the challenger generates $K_{\mathcal{S}}$ uniformly at random in case $\mathcal{U}$ is honest. For a Send query to a partner $\mathcal{U}$ instance on an oracle-generated tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, \tau_{\mathcal{S}}, hp_{\mathcal{S}})$, the challenger sets $K_{\mathcal{U}} \leftarrow K_{\mathcal{S}}$.

Distinguishing this game from the previous can be reduced to the security of KDF w.r.t. uniformly random sources.

Denoting by $\varepsilon_{\mathsf{KDF}}(0)$ the supremal advantage of any efficient adversary which makes no oracle query in the KDF security game with scheme KDF (cf. Appendix E), this game can be distinguished from the previous one with an advantage of at most $|I|^2\varepsilon_{\mathsf{KDF}}(0)$.

Game 14. In this game, if $\mathcal{U}$ is honest, the challenger answers Send queries to $\mathcal{S}$ instances on $\left(\tilde{C}_M, aux, \tau_{\mathcal{U}}\right)$ tuples by generating $M_{\mathcal{S}} \leftarrow_{\$} \mathbb{G}_1$. This game is perfectly indistinguishable from the previous one as $H_{\mathcal{S}}$ is uniformly random in the latter.

Game 15. To answer Send queries to $\mathcal{T}$ instances on oracle-generated pairs $(C_{\mathcal{U}}, hp_{\mathcal{U}})$, the challenger now generates $K \leftarrow_{\$} \mathbb{G}_1$ and computes $C_{\mathcal{T}}$ as

$\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}(ek', K)$. To answer Send queries to the partner $\mathcal{S}$ instance in this session on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, the challenger verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = K$. The indistinguishability from the previous game follows from the IND-PCA security of $\mathcal{E}_{\mathsf{pca}}$. Indeed, for queries to $\mathcal{S}$ instances as above, if $(C_{\mathcal{T}}, ovk)$ is oracle-generated (i.e., $(*, ovk, *, C_{\mathcal{T}}, *)$ was computed as an answer to a Send query), then the reduction need not make a decryption query as it computed it itself. If $(C_{\mathcal{T}}, ovk)$ is adversarially generated, i.e., $C_{\mathcal{T}}$ was computed with a label different from ovk, the reduction algorithm can then make a decryption query.

Distinguishing this game from the previous one can thus be done with an advantage of at most $q_{\mathsf{Send}}\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})$.

Game 16. To answer Send queries to $\mathcal{T}$ instances on oracle-generated pairs $(C_{\mathcal{U}}, hp_{\mathcal{U}})$, the challenger now generates $K, K' \leftarrow_{\$} \mathbb{G}_1$ and computes $C_{\mathcal{T}}$ as $\mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{ovk}(ek', K)$. To answer Send queries to the partner $\mathcal{S}$ instance in this session on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if $(C_{\mathcal{T}}, ovk)$ is oracle-generated in the same session, the challenger skips the verification, otherwise the challenger verifies that $\mathcal{E}_{\mathsf{pca}}.\mathsf{Dec}^{ovk}(dk', C_{\mathcal{T}}) = K'$. Once again, indistinguishability from the previous game follows from the IND-PCA security of $\mathcal{E}_{\mathsf{pca}}$. Adversary $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $q_{\mathsf{Send}}\varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}})$.

Game 17. For Send queries to $\mathcal{S}$ instances on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if ovk and $\pi_{\mathcal{T}}$ are oracle-generated but in different sessions, return $\perp$. Denoting by $\varepsilon_{\mathsf{OTS}}$ the supremal advantage of any efficient adversary in the strong one-time security game, $\mathcal{A}$ can distinguish this game fromt the previous with an advantage of at most $q_{\mathsf{Send}}|I|\varepsilon_{\mathsf{OTS}}$ (the reduction algorithm must guess the $\mathcal{T}$ instance for which it sets the one-time signing key as that of the challenger).

Game 18. For Send' queries to $\mathcal{S}$ instances on tuples $(C_{\mathcal{U}}, hp_{\mathcal{U}}, ovk, \tilde{\pi}_{\mathcal{T}}, C_{\mathcal{T}}, \sigma_{\mathcal{T}})$, if ovk is adversarially generated and $\pi_{\mathcal{T}}$ is oracle-generated, the challenger returns $\perp$. Adversary $\mathcal{A}$ can distinguish this game from the previous one with an advantage of at most $1/p$. Indeed, if $\pi_{\mathcal{T}}$ is oracle-generated, there exists a K' as defined in Game 16 which is independent of all the other messages computed by the challenger and of all the inputs from $\mathcal{A}$.

Winning the 1-out-of-2 RCCA game can then be reduced to winning the last game as follows. At the beginning of the game, the reduction algorithm, denoted $\mathcal{B}$, guesses again the identities $\mathcal{U}^*$, $\mathcal{T}^*$ and $\mathcal{S}^*$. (If $\mathcal{A}$ later makes its Test query on a different tuple of identities, $\mathcal{B}$ simply aborts and sends to the challenger a bit chosen uniformly at random.) $\mathcal{B}$ then sets the public key as the public key of $\mathcal{U}^*$ and asks for a secret key share that it sets as the share of $\mathcal{S}^*$. The other share is then implicitly the share of $\mathcal{T}^*$.

Recall that if $\mathcal{A}$ corrupts $\mathcal{U}^*$, algorithm $\mathcal{B}$ can reply to Send queries to $\mathcal{T}^*$ instances with $\perp$ and perfectly emulate the challenger of the last game.

Whenever $\mathcal{A}$ makes its first Send query to an $\mathcal{S}^*$ instance on a valid tuple $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*}, ovk, \tilde{\pi}_{\mathcal{T}^*}, C_{\mathcal{T}^*}, \sigma_{\mathcal{T}^*})$ such that $(C_{\mathcal{U}^*}, hp_{\mathcal{U}^*})$ is oracle-generated but

$(C_{\mathcal{T}^*}, ovk)$ is adversarially generated, ovk and $\pi_{\mathcal{T}^*}$ are necessarily adversarially generated by definition of the challenger of the last game. By CM simulation soundness of $\mathsf{SS\_GS}$ w.r.t. additive transformations (which holds under the existential unforgeability of SIG), either (i) $f\left(pk_1, \mathbf{c}(\alpha_{1,\mathcal{T}^*})\right) f\left(g_1, \mathbf{c}(\alpha_{2,\mathcal{T}^*})\right) = f\left(pk_2 pk_1^{-\alpha_{1,\mathcal{S}^*}} g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}\right) f(\Theta, \mathbf{v})$, or there exists an oracle-generated proof (respectively by an $\mathcal{S}^*$ instance or by a $\mathcal{T}^*$ instance) $(\mathbf{c}_1, \mathbf{c}_2, \Theta')$ such that (ii-a) $f(pk_1, \mathbf{c}_1) f(g_1, \mathbf{c}_2) = f(pk_2, \mathbf{u}) f(\Theta', \mathbf{v})$ or (ii-b) $f(pk_1, \mathbf{c}_1) f(g_1, \mathbf{c}_2) = f(pk_2 pk_1^{-\alpha_{1,\mathcal{S}^*}} g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}) f(\Theta', \mathbf{v})$ and a tuple $(\mathbf{c}(\zeta_1), \mathbf{c}(\zeta_2), \Theta'')$, with $(\zeta_1, \zeta_2) \in \mathbb{Z}_p^2$ representing a transformation, such that $f(pk_1, \mathbf{c}(\zeta_1)) f(g_1, \mathbf{c}(\zeta_2)) = f\left(pk_2 pk_1^{-\alpha_{1,\mathcal{S}^*}} g_1^{-\alpha_{2,\mathcal{S}^*}}, \mathbf{u}\right) f(\Theta'', \mathbf{v})$. It follows that for $i = 1, 2$ either $\mathbf{c}(\alpha_{i,\mathcal{T}^*})$ or $\mathbf{c}(\zeta_i)$ is a commitment to $\alpha_{i,\mathcal{T}^*}$; and assume without loss of generality that it is the former. $\mathcal{B}$ then computes $\mathbf{c}(\alpha_i) \leftarrow \mathbf{c}(\alpha_{i,\mathcal{T}^*}) \mathbf{u}^{\alpha_{i,\mathcal{S}^*}}$ for $i = 1, 2$, i.e., commitments to α_i with the same randomness used to compute $\mathbf{c}(\alpha_{i,\mathcal{T}^*})$.

If $\mathcal{A}$ makes a Test query on the guessed identities, $\mathcal{B}$ simply forwards (M_0, M_1) to the RCCA challenger, and receives back a challenge ciphertext C^* which encrypts M_b for $b \leftarrow_\$ \{0, 1\}$. (If the guess was incorrect, $\mathcal{B}$ aborts its interaction with $\mathcal{A}$ and sends a uniformly random bit to the challenger.) Note that $x_3^* M_b^{-1} = {x_1^*}^{\alpha_1} {x_2^*}^{\alpha_2}$ and that $\mathrm{dlog}_{pk_1} x_1^* = \mathrm{dlog}_{g_1} x_2^* = \mathrm{dlog}_{pk_2} x_3^*$. Therefore, $f\left(x_1^*, \mathbf{c}(\alpha_1)\right) f\left(x_2^*, (\alpha_2)\right) = f\left(x_3^* M_b^{-1}, \mathbf{u}\right) f(\Theta, \mathbf{v})$. Algorithm $\mathcal{B}$ can then test the previous equality with M_0 and M_1 and win the 1-out-of-2 RCCA game with at least the same advantage as $\mathcal{A}$ in the selective P-IND-RCCA game.

Consequently, in case 2), $\mathcal{A}$ wins the P-IND-RCCA game with an advantage of at most

$$\varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Exec}}) + \varepsilon^{zk}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + q_{\mathsf{Exec}} \varepsilon^{\mathrm{ind-cpa}}_{\mathcal{E}_{\mathsf{pca}}} + |I| \varepsilon_{\mathsf{OTS}} + 2|I|^2 \varepsilon_{\mathsf{KDF}}(0)$$
$$+ 2 q_{\mathsf{Send}}(q_{\mathsf{Send}} + 1) \varepsilon^{\mathrm{ind-pca}}_{\mathcal{E}_{\mathsf{pca}}}(q_{\mathsf{Send}}) + \varepsilon^{\mathrm{sound}}_{\mathsf{SS\_GS}}(q_{\mathsf{Send}}) + |U||T||S| \mathsf{Adv}^{\mathrm{ind-rcca}}_{\mathsf{G}, \mathcal{E}_{\mathsf{rcca}}}(\lambda).$$

□

Theorem F.2 (Blindness). *$\mathcal{E}$ satisfies blindness under the SXDH assumption over G.*

Proof. The blindness property of $\mathcal{E}$ can be proved as follows via a sequence of indistinguishable games starting from the real game and ending with a game in which the advantage of any adversary is nil.

Game 0. This is the real game.

Game 1. The challenger generates GS parameters $\mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{w}$ in witness-indistinguishability mode. This CRS is computationally indistinguishable from the one in the previous game under the SXDH assumption over G.

Game 2. Instead of computing $\mathbf{c}(R)$ and Σ as algorithm Blind, the challenger simulates those values with the GS proof-system simulator. Further denote the resulting algorithm as $\mathsf{SimBlind}$. Game 2 is perfectly indistinguishable from Game 1 since the simulation is perfect.

Game 3. In this game, the challenger runs $\mathsf{SimBlind}$ on $\mathsf{Enc}(pk, \mathsf{Dec}(sk, C_b))$. Game 3 is perfectly indistinguishable from Game 2 since the scheme of Faonio et al. is perfectly unlikable.

Game 4. Instead of running algorithm $\mathsf{SimBlind}$ on $\mathsf{Enc}(pk, \mathsf{Dec}(sk, C_b))$, the challenger runs it on $\mathsf{Enc}(pk, K)$ for $K \leftarrow_{\$} \mathbb{G}_T$. Game 4 is perfectly indistinguishable from Game 3 since $\mathsf{SimBlind}$ computes $\tilde{x}_3$ as $x_3 R$ for $R \leftarrow_{\$} \mathbb{G}_T$, which entirely re-randomizes $\mathsf{Dec}(sk, C_b)$ in Game 3.

Note that in Game 4, the advantage of any adversary is nil.

It follows that $\mathcal{E}$ satisfies blindness under the SXDH assumption over G. □

Theorem F.3 (Verifiability). *$\mathcal{E}$ is verifiable if $\{h_\kappa\}_\kappa$, $\{H_\kappa\}_\kappa$, g. and $\hat{g}$. satisfy the assumptions above and if the SXDH assumption over G holds.*

Proof. Suppose that there exists an efficient adversary $\mathcal{A}$ which wins the verifiability game with a non-negligible probability, i.e., it returns $p_{\mathcal{U}}$ and C such that the honest execution of U on $(pk, p_{\mathcal{U}}, C)$ with $\mathcal{A}$ results in a value different from $\mathsf{Dec}(sk, C)$ and $\perp$. In the event in which $\mathcal{A}$ wins the game, then either $\mathsf{Dec}(dk, C_{\mathcal{S}}) = p_{\mathcal{U}}$ or not. If so, then there exists an algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as sub-routine and contradicts the perfect soundness of the GS proof system. If $\mathsf{Dec}(dk, C_{\mathcal{S}}) \neq p_{\mathcal{U}}$, then there exists an algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as sub-routine and wins the MAC game with non-negligible probability.

In the first case, the correctness of the SPHF guarantees that $\mathsf{Hash}\left(hk_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{U}}}, C_{\mathcal{S}}\right) \mathsf{ProjHash}\left(hp_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{U}}}, \perp, r_{\mathcal{U}}\right) = \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}\right) \mathsf{ProjHash}\left(hp_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp, r_{\mathcal{S}}\right)$. Therefore, if the value returned by U at the end of the protocol is different from both $\mathsf{Dec}(sk, C)$ and $\perp$, adversary $\mathcal{A}$ necessarily contradicted the soundness of $\mathsf{SS\_GS}$ for to the language

$$
\begin{aligned}
\big\{(ek, pk_{\mathcal{U}}, C_{\mathcal{U}}, C_{\mathcal{S}},\ hp_{\mathcal{U}}, \tilde{C}_M, M_{\mathcal{S}}) \colon \exists\, (\alpha_i, hk_{\mathcal{S}}, p_{\mathcal{S}}, r_{\mathcal{S}}),\ & \\
pk_2 &= pk_1^{\alpha_1} g_1^{\alpha_2} \\
hp_{\mathcal{S}} &= \mathsf{ProjKG}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp\right) \\
C_{\mathcal{S}} &= \mathcal{E}_{\mathsf{pca}}.\mathsf{Enc}^{hp_{\mathcal{S}}}(ek, p_{\mathcal{S}}; r_{\mathcal{S}}) \\
M_{\mathcal{S}} \tilde{x}_3^{-1} &= \tilde{x}_1^{-\alpha_1} \tilde{x}_2^{-\alpha_2} \mathsf{Hash}\left(hk_{\mathcal{S}}, \mathcal{L}_{p_{\mathcal{S}}}, C_{\mathcal{U}}\right) \\
&\quad \cdot \mathsf{ProjHash}\left(hp_{\mathcal{U}}, \mathcal{L}_{p_{\mathcal{S}}}, \perp, r_{\mathcal{S}}\right)\big\},
\end{aligned}
$$

which is impossible under the existential unforgeability of Jutla and Roy's signature, which relies on the SXDH assumption.

In the second case, i.e., if $\mathsf{Dec}(dk, C_{\mathcal{S}}) \neq p_{\mathcal{U}}$, the verifiability of $\mathcal{E}$ can be reduced to the security of the MAC through a sequence of games as below.

Game 0. This is the real game.

Game 1. In this game, the challenger replaces $H_{\mathcal{U}}$ with a uniformly random value. By the smoothness of the SPHF, Game 1 is perfectly indistinguishable from Game 0.

Game 2. The challenger now generates a random key $K_{\mathcal{U}}$ instead of computing it with the KDF. The indistinguishability of Game 2 from Game 1 can then be reduced to the security of KDF w.r.t. uniformly random source.

Note that as the distribution of the source is independent of the adversary, generating the salt value XTS before the end of the PAKE does not raise any issue in the reduction. Indeed, the reduction algorithm can generate a uniformly random source value at the beginning of the reduction, submit it to the KDF-security-game challenger, then receive back a uniform salt value before generating the other parameters for $\mathcal{E}$.

Therefore, under the assumptions on the compression functions which imply the security of Krawczyk's KDF, an efficient adversary can distinguishing Game 2 from Game 1 with advantage at most $\varepsilon_{\mathsf{KDF}}(q_{\mathsf{Send}})$, with $\varepsilon_{\mathsf{KDF}}(q_{\mathsf{Send}})$ denoting the supremal advantage of any efficient adversary which makes at most q_{Send} queries in the KDF security game with scheme KDF (cf. Appendix E).

As $\mathcal{A}$ must compute a tuple $(\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}}, \tau_{\mathcal{S}})$ such that $\mathsf{MAC}\,(K_{\mathcal{U}}, (\tilde{\pi}_{\mathcal{S}}, C_{\mathcal{S}}, hp_{\mathcal{S}})) = \tau_{\mathcal{S}}$ to win Game 0 without any prior MAC computation by U, it does so with non-negligible probability in Game 2. It follows that $\mathcal{A}$ can then be run as sub-routine to win the MAC game with non-negligible probability. Once again, under the assumptions on the compression functions (which imply the security of HMAC), an efficient adversary can only do so with negligible probability; a contradiction. It follows that such an adversary $\mathcal{A}$ cannot exist and $\mathcal{E}$ is thus verifiable. □